We Together

Community Development App

By

ABRAHAM RICHARD (2241104)

GREESHMA GIRISH C (2241130)

SANDEEP MATHEW (2241155)

Under the supervision of

Dr. K. Saravanakumar

Project Report Submitted in partial fulfilment of the requirements

of V semester Bachelor of Computer Applications,

CHRIST (Deemed to be University)

October - 2024

CERTIFICATE

This is to certify that the report titled ***We Together*** *is a bona fide record of work done by* ***Abraham Richard (2241104), Greeshma Girish C (2241130)*** *and* ***Sandeep Mathew (2241155)*** *of Christ University, Bengaluru, in partial fulfillment of the requirements of 5th Semester BCA during the year 2024.*

Head of the Department **Project Guide**

Valued-by:

1.

2.

Name : Sandeep Mathew
Register Number : 2241155
Examination Centre : Christ University
Date of Exam :

ACKNOWLEDGEMENTS

First of all, we thank God almighty for his immense grace and blessings showered on us at everystage of this work. We are grateful to our respectable Head, Department of Computer Science, CHRIST (Deemed to be University), **Dr. Ashok Immanuel V**, for providing the opportunity to take up this project as part of my curriculum.

We also pay our gratitude to the Coordinator, Department of Computer Science, CHRIST (Deemed to be University) **Dr. Sagaya Aurelia P** for their support throughout.

We are grateful to our guide, **Dr. K. Saravanakumar** Associate Professor, Department of Computer Science, CHRIST (Deemed to be University), whose insightful leadership and knowledge benefited us to complete this project successfully. Thank you so much for your continuous support and presence whenever needed.

We would also like to thank our Alumni evaluator **Mr. Sanju C Paul,** whose knowledge and guidance benefited us in making the changes as per the industry requirement. Thank you so much for your support and presence whenever needed.

We express our sincere thanks to all faculty members and staff of the Department of Computer Science, CHRIST (Deemed to be University), for their valuable suggestions during the course ofthis project. Their critical suggestions helped us to improve the project work.

Last but not the least, we would like to thank everyone who is involved in the project directly or indirectly.

ABSTRACT

We Together aims to tackle the pressing challenges of gender inequality and economic disparity by creating a strong Self-Help Group (SHG) platform that places women's empowerment at its core while inclusively engaging men. The initiative is dedicated to cultivating a supportive ecosystem where women can develop entrepreneurial skills, gain access to essential financial resources, and actively participate in decision-making processes. By fostering an environment of collaboration and mutual support, "We Together" harnesses the collective strength of SHGs to drive economic independence and social cohesion.

Through this initiative, women are empowered to break barriers and establish sustainable livelihoods, transforming themselves into key contributors to their communities and the economy. "We Together" also encourages male participation, promoting a shared responsibility in building a more inclusive society. The platform's focus extends beyond economic growth; it addresses societal norms and structures, creating opportunities for women to lead and innovate.

Aligning with the United Nations Sustainable Development Goals, "We Together" directly supports SDG Goal 5 (Gender Equality) by empowering women to take control of their futures, and SDG Goal 9 (Industry, Innovation, and Infrastructure) by fostering innovative solutions that drive inclusive economic development. Ultimately, the initiative envisions a future where everyone, regardless of gender, has the opportunity to thrive.

Table of Contents

List of Tables

List of Figures

1.INTRODUCTION

1.1 Overview of the system

In the current socio-economic landscape, women face significant challenges in accessing equal job opportunities and economic empowerment compared to men. Despite efforts to promote gender equality, women still encounter barriers such as limited access to resources, skills training, and entrepreneurial support. Concurrently, men also seek avenues for community support and economic advancement.

1.2 Functionalities of System

1.2.1 Job Management: Admins have the ability to create jobs by entering essential details such as the job name, description, time and date, and location. They can also define the profit distribution for production-related jobs and include contact information for the responsible person. Once jobs are created, admins can view a comprehensive list of all jobs and manage them effectively. If a job is no longer required or has been completed, admins have the option to delete it.

1.2.2 Loan Management: The Loan Management feature provides a detailed overview of all loans taken by users within the We Together community. It includes specific details such as the loan amount, the due date for repayment, and a record of the total amount lent by We Together from the bank. This comprehensive tracking ensures transparency and accountability, making it easier for administrators to monitor outstanding loans and manage financial resources efficiently.

1.2.3 Attendance Management: The Attendance Management feature enables administrators to efficiently track the participation of We Together community members during events, meetings, and other gatherings. Admins can mark attendance for each member, ensuring accurate records of their involvement. The app also allows for easy updates to attendance data, making it straightforward to reflect any changes or corrections.

2.SYSTEM ANALYSIS

2.1 Existing Systems

The concept of a self-help group (SHG) management platform is still emerging, but several systems already exist in the market that provide similar services. Existing solutions in this domain focus on promoting sustainability within SHGs and enhancing their practices. They place a strong emphasis on empowering individuals within these groups by providing them with the necessary tools and support.

The following are a few projects that the team has worked on improving (along with the hyperlink)

LokOS App

LokOS is a mobile application that empowers communities and individuals to engage more actively in democratic processes and decision-making at the local level. (LokOS) [1]

Kudumbashree Website

Kudumbashree is essentially a community network that covers the entire State of Kerala. It consists of a three-tier structure with Neighborhood Groups (NHGs) as primary level units, Area Development Societies (ADS) at the ward level, and Community Development Societies (CDS) at the local government level. (Kudumbashree) [2]

2.1.1 Limitation of Existing Systems

Lok OS App:

- **Absence of loan management system:** One significant drawback of the Lok OS app is the lack of a comprehensive loan management system. It may lead to administrative challenges, difficulties in ensuring timely loan repayments .

- **Missing admin console:** The Lok OS app does not have an admin console system, which is essential for efficient system administration, user management.

Kudumbashree website

- **Usability issues:** The official website of Kudumbashree.org suffers from usability issues as the user interface is not intuitive and user-friendly. This can result in a suboptimal browsing experience for users trying to access information or navigate through the website.
- **Lack of login features:** The website lacks essential login features, such as user accounts or member logins. This absence of login functionality restricts user engagement and interaction with personalized content and services, limiting the website's overall functionality.

2.2 Proposed System

The proposed system for the "We Together" app is a comprehensive platform designed to assist Self-Help Group (SHG) members in tracking and managing their activities more effectively. It centralizes data related to various community initiatives, including job opportunities, financial transactions, attendance management, providing users with valuable insights to enhance their participation and decision-making. This system aims to streamline processes, support entrepreneurship, and promote collaborative growth within the community. By fostering transparency and efficiency, the proposed platform empowers SHG members to develop sustainable livelihoods and contributes to the broader goal of social and economic empowerment.

2.2.1 Benefits of Proposed System

The proposed "We Together" system significantly reduces paperwork by digitizing the management of job opportunities, financial transactions, and attendance. This not only simplifies processes but also provides SHG members with quick access to essential data, improving decision-making and collaboration. By minimizing manual tasks, the platform promotes efficiency, transparency, and sustainable growth, empowering members to focus on developing their livelihoods.

2.3 Requirements Specifications

Sources of Requirements	Requirements specification
User Needs and Feedback : Requirements can be derived from direct interactions with SHG members, community feedback sessions, surveys, and focus group discussions. Understanding user needs and preferences is crucial to ensure the app meets their expectations.	Job Creation : Admins can create jobs with details such as name, description, time and date, location, profit distribution, and contact information.
Organizational Policies : Aligning the app's functionality with the internal policies and operational procedures of the SHG and its partner organizations.	Loan Tracking : Maintain details of users who took loans, including loan amount, due date, and total amount lent.
Market Trends and Competitor Analysis: Studying similar apps and market trends to identify features that are in demand and ensuring "We Together" offers competitive and innovative solutions.	Listing and Selling: Facilitate buying and selling of goods and services created by SHG members. Include secure payment options, ratings and reviews, and detailed listings.

End Users and Consumers: Understanding the needs and concerns of end users and consumers can provide insights into how to design a user-friendly application that meets their expectations and encourages active participation in the community.	User-Friendly Mobile App: Develop a user-friendly mobile app or web portal that allows users to easily navigate through various features, including job management, loan tracking, attendance, notifications, forums, and the marketplace.
Technical Standards and Best Practices: Following industry standards and best practices for software development, data security, user interface design, and performance optimization will ensure the app is robust, secure, and user-friendly	Mark Attendance: Admins can mark attendance for community members during events or meetings. Admins can view and update attendance data for each member.

2.3.1 Functional Requirement

Req. No	Requirement	Specific Example
FR01	User Interaction: How users will interact with the system, what actions they can perform, and what outputs they can expect.	Users should be able to log in, view job listings, apply for jobs, and receive notifications about job statuses. Admins should be able to create, edit, and delete job listings.
FR02	Data Processing: How data will be collected, stored, processed, and presented within the system.	Authorized personnel should be able to access the sensor data remotely through a secure web interface or mobile application.
FR03	System Behaviour: Describing how the system should respond to various inputs and perform different tasks.	The system must securely store loan details including loan amounts, due dates, and repayment status. Admins should be able to access and update this information as needed.
FR04	System Components: Identifying the various modules, components, or features that make up the system.	The system must allow admins to create and manage job listings, including specifying details such as job name, description, time, date, location, profit distribution, and contact details.
FR05	Performance: Defining performance metrics such as response times, processing speed, and capacity to handle concurrent users.	The system should be capable of handling up to 500 concurrent users without any errors.

FR06	Security: Specifying the measures that ensure data security, access control, and protection against unauthorized access.	The system should ensure that only authorized users have access to admin functionalities, with different levels of access control based on user role.
FR07	Integration: Addressing how the system will integrate with other systems.	The system should integrate with external calendar services to synchronize event schedules and attendance records.

2.4. System Requirements

2.4.1 Hardware Requirements	2.4.2 Software Requirements	2.4.3 Network Requirements
RAM • 2 GB Internal Storage • 4 GB or higher Processor • ARMv7 (32-bit). ARM64 (64-bit). x86, x86_64 or higher	Operating System • Android 7 or higher	Wi-Fi: Internet Required

3.SYSTEM DESIGN

3.1 Block Diagram

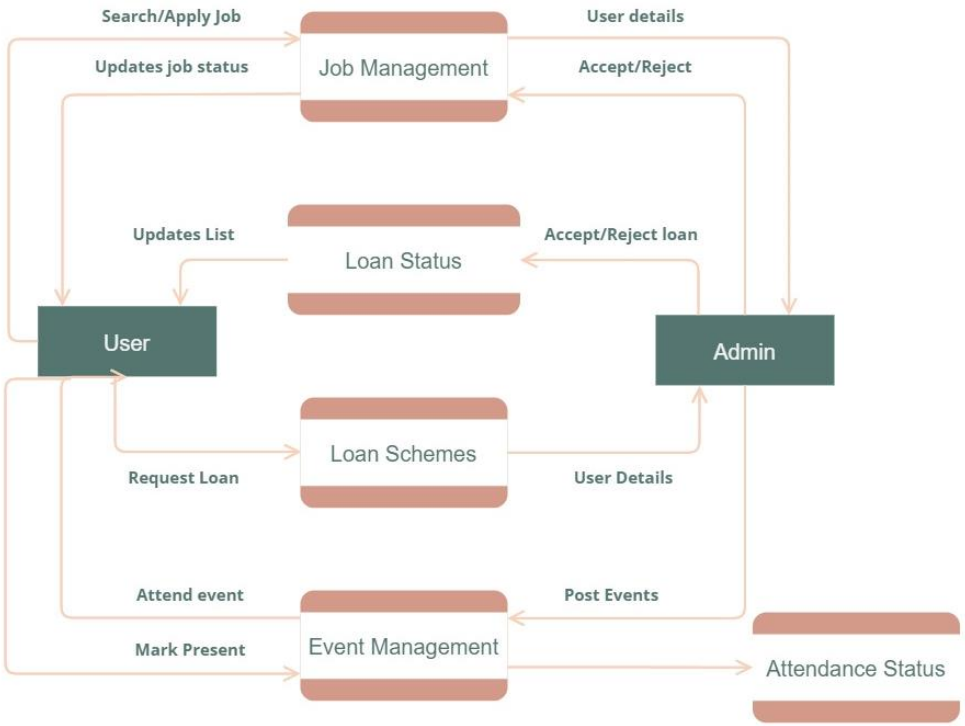

Fig 3.1 Block diagram showing graphical representation of the system

3.2 Database Design

3.2.1 Database Tables

Event Table

Table No.1 Event Table Data Dictionary

Attributes	DataTypes
_id	Object()
title	String
description	String
venue	String
place	String
date	Date
time	String

Event Attendance Table

Table No.2 Event Attendance Table Data Dictionary

Attributes	Data Types
_id	Object()
name	String
email	String
eventTitle	String
status	String

Job Table

Table No.3 Job Table Data Dictionary

Attributes	Data Types
_id	Object()
title	String
level	String
salary	Number
location	String
type	String

Job Applicants Table

Table No.4 Job Applicants Table Data Dictionary

Attributes	Data Types
_id	Object()
name	String
email	String
phone	String
address	String
aadhaar	String
jobtitle	String

Loan Table

Table No.5 Loan Table Data Dictionary

Attributes	Data Types
_id	Object()
title	String
amount	Number
description	String
interest	String

Loan Applicants Table

Table No.6 Loan Applicants Table Data Dictionary

Attributes	Data Types
_id	Object()
name	String
email	String
phone	String
address	String
aadhaar	String
loantitle	String
status	String

User Table

Table No.7 User Table Data Dictionary

Attributes	Data Types
_id	Object()
name	String
user_name	String
password	String
email	String
contact no	Number
gender	String
isAdmin	Boolean

Forum Table

Table No.8 Forum Table Data Dictionary

Attributes	Data Types
_id	Object()
user	String
content	String
image	String
likes	Array

3.3 ER Diagram

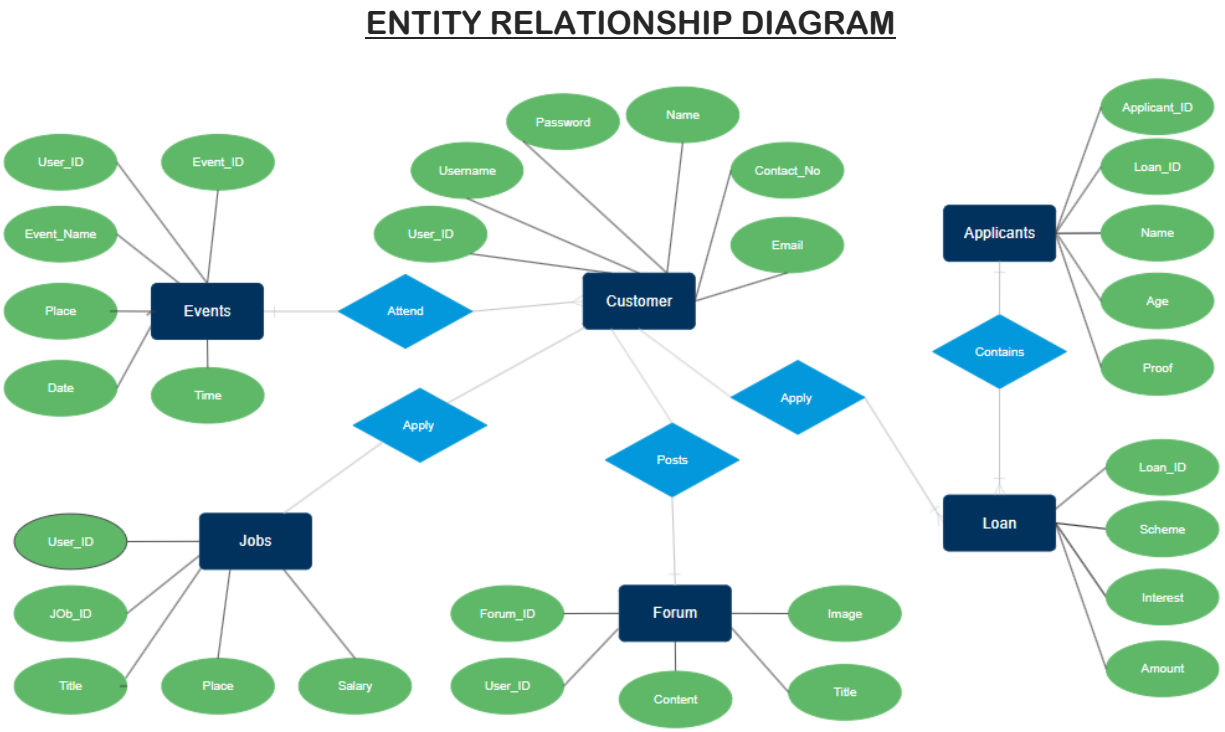

Fig 3.3 Entity Relationship diagram

3.4 Class Diagram

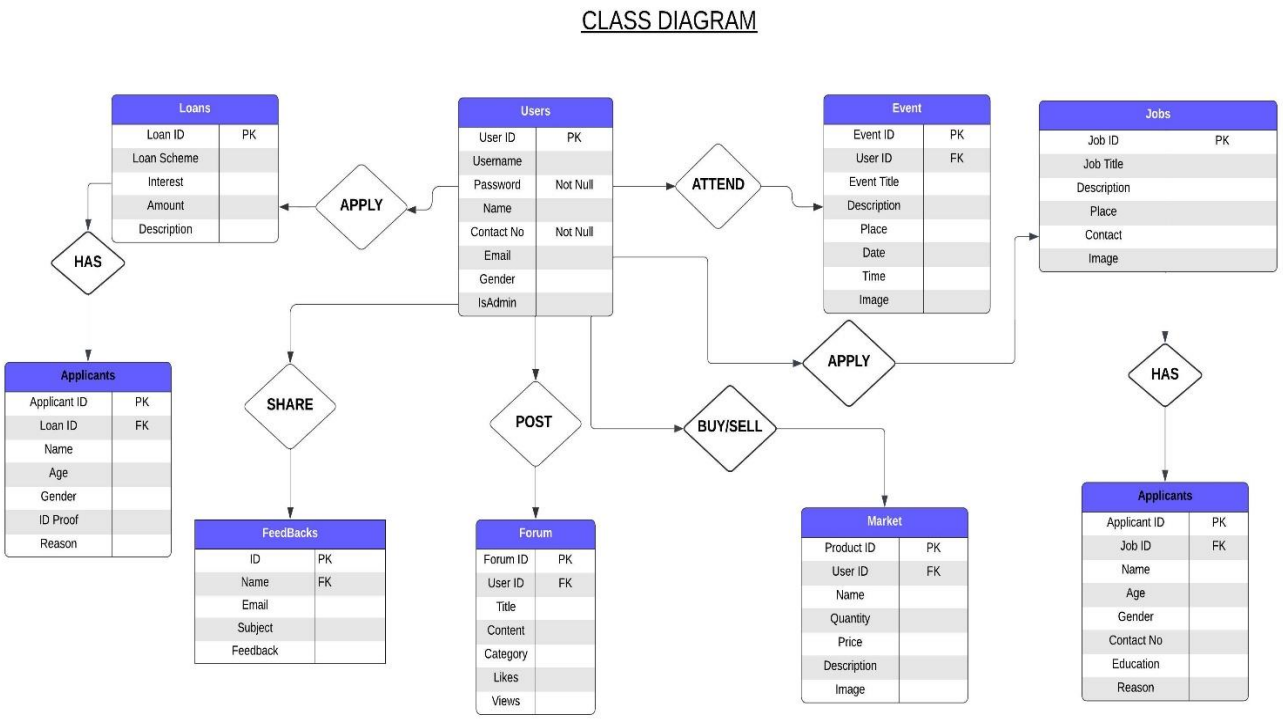

Fig 3.4 Class diagram

3.5 Data Flow Diagram

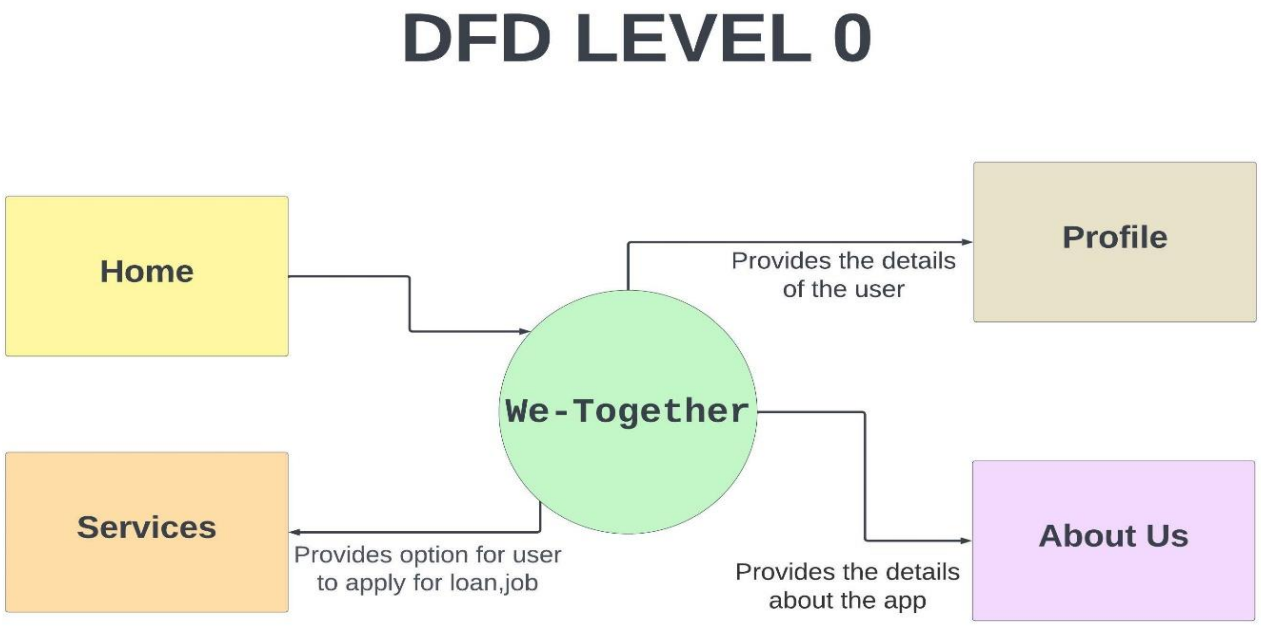

Fig 3.5 Data Flow Diagram Level-0 depicting overview of the entire system

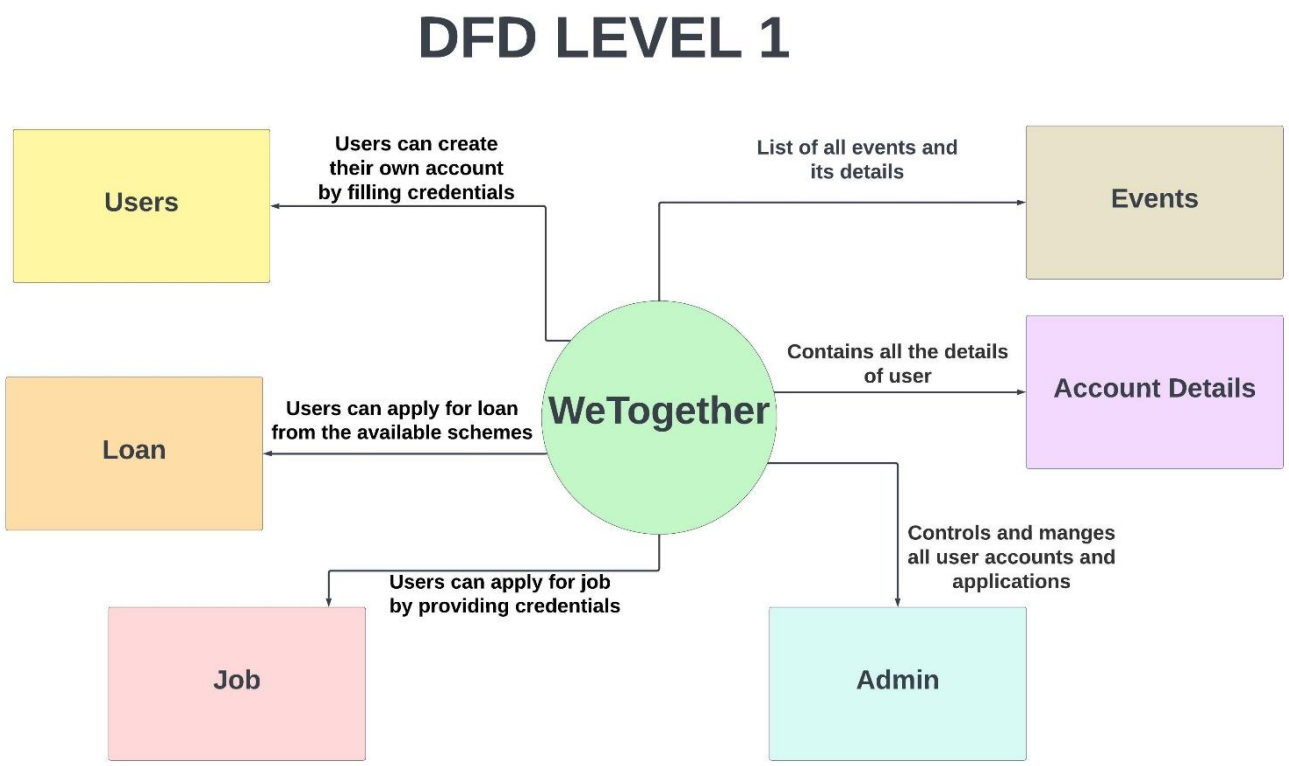

Fig 3.5 Data Flow Diagram Level-1 depicting overview of the entire system

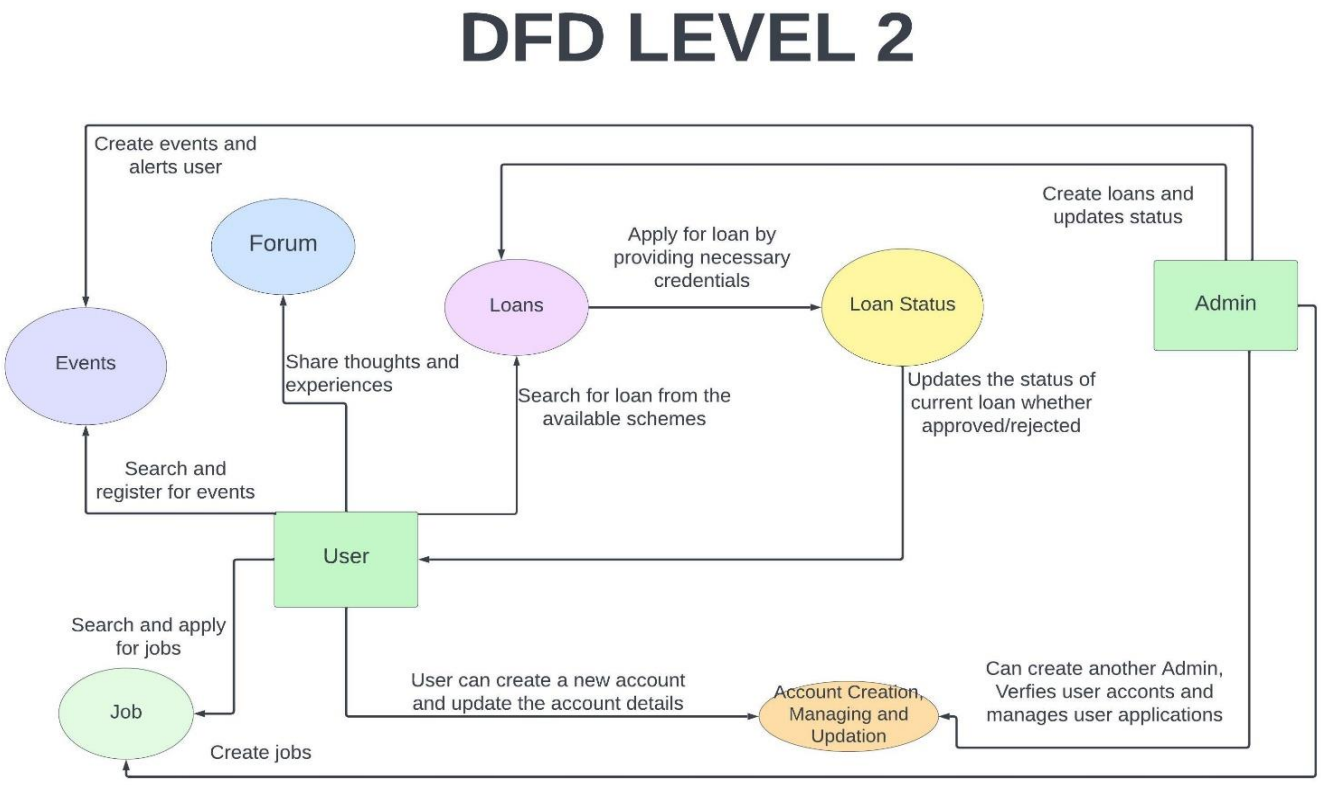

Fig 3.5 Data Flow Diagram Level-2 depicting overview of the entire system

3.6 Use Case Diagram

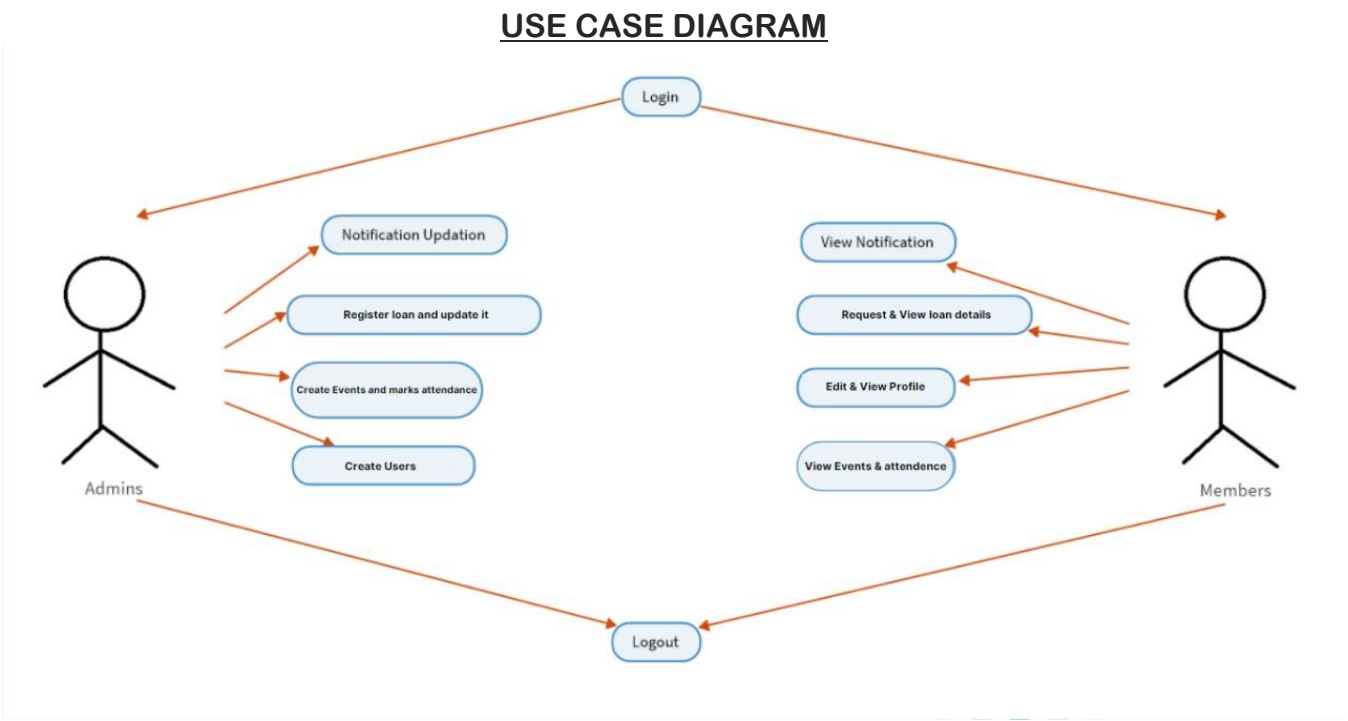

Fig 3.6 Use Case diagram for Users and Admin

4.IMPLEMENTATION

4.1 Source Code

App.js

```
import React, { useEffect, useState } from 'react';
import { StatusBar } from 'expo-status-bar';
import { createDrawerNavigator } from '@react-navigation/drawer';
import { createStackNavigator } from '@react-navigation/stack';
import { NavigationContainer } from '@react-navigation/native';
import * as SplashScreen from 'expo-splash-screen';
import { useFonts } from 'expo-font';

// Import components
import AboutUs from './components/UserComponents/AboutUs';
import CustomDrawerContent from './components/UserComponents/CustomDrawerContent';
import HomeScreen from './components/UserComponents/HomeScreen';
import ProfileScreen from './components/UserComponents/ProfileScreen';
import EventAttendance from './components/UserComponents/EventAttendance';
import LoanStatus from './components/UserComponents/LoanStatus';
import LoanRequest from './components/UserComponents/LoanRequest';
import JobList from './components/UserComponents/Jobs';
import Forum from './components/UserComponents/Forum';
import HomeScreenAdmin from './components/AdminComponents/HomeScreenAdmin';
import Login from './components/UserComponents/login';
import SignUp from './components/UserComponents/SignUp';
import Loans from './components/AdminComponents/Loans';
import AdminJobs from './components/AdminComponents/AdminJobs';
import AdminEvents from './components/AdminComponents/Events';


const Drawer = createDrawerNavigator();
const Stack = createStackNavigator();

export default function App() {
  const [fontsLoaded] = useFonts({
    'Poppins-Bold': require('./fonts/Poppins-Bold.ttf'),
    'Poppins-Regular': require('./fonts/Poppins-Regular.ttf'),
    'Poppins-SemiBold': require('./fonts/Poppins-SemiBold.ttf'),
    'Poppins-Normal': require('./fonts/Poppins-Regular.ttf'),
    'Poppins-LightBold':require('./fonts/Poppins-SemiBold.ttf')
  });

  const [isAuthenticated, setIsAuthenticated] = useState(false);
  const [isAdmin, setIsAdmin] = useState(false);
```

```
useEffect(() => {
  async function prepare() {
    await SplashScreen.preventAutoHideAsync();
  }
  prepare();
}, []);

useEffect(() => {
  if (fontsLoaded) {
    SplashScreen.hideAsync();
  }
}, [fontsLoaded]);

if (!fontsLoaded) {
  return null; // Show nothing until fonts are loaded
}

const handleLoginSuccess = () => {
  setIsAuthenticated(true);
  setIsAdmin(false); // Set user as non-admin after login success
};

const handleAdminLogin = () => {
  setIsAuthenticated(true);
  setIsAdmin(true); // Set user as admin after admin login
};

const handleLogout = () => {
  console.log('Logging out...');
  setIsAuthenticated(false);
  setIsAdmin(false); // Reset the admin status on logout
};

return (
  <NavigationContainer>
    {isAuthenticated ? (
      <Drawer.Navigator
        initialRouteName={isAdmin ? 'HomeAdmin' : 'Home'}
        drawerContent={(props) => (
          <CustomDrawerContent {...props} onLogout={handleLogout} />
        )}
        screenOptions={{
          headerShown: true,
          headerStyle: { backgroundColor: '#5A67D8' },
          headerTintColor: '#fff',
        }}
      >
        {isAdmin ? (
          <>
```

```
            <Drawer.Screen name="HomeAdmin">
              {(props) => <HomeScreenAdmin {...props} handleLogout={handleLogout}
/>}
            </Drawer.Screen>
            <Drawer.Screen name="Loans" component={Loans} />
            <Drawer.Screen name="Jobs" component={AdminJobs} />
            <Drawer.Screen name="Events" component={AdminEvents} />
          </>
        ) : (
          <>
            <Drawer.Screen name="Home" component={HomeScreen} />
            <Drawer.Screen name="Profile" options={{
                headerShown: true,
                headerStyle: { backgroundColor: '#024578' },
                headerTintColor: '#fff',
                headerTitleAlign:'left',
              }} >
            {(props) => <ProfileScreen {...props} handleLogout={handleLogout}  />}
            </Drawer.Screen>
            <Drawer.Screen name="Events" component={EventAttendance} />
            <Drawer.Screen name="Jobs" component={JobList} />
            <Drawer.Screen name="Loan Request" component={LoanRequest} />
            <Drawer.Screen name="Loan Status" component={LoanStatus} />
            <Drawer.Screen  name="Forum"  component={Forum}
              options={{
                headerShown: true,
                headerStyle: { backgroundColor: '#024578' },
                headerTintColor: '#fff',
                headerTitleAlign:'left',
              }}
            />

            <Drawer.Screen name="About Us" component={AboutUs} />
          </>
        )}
      </Drawer.Navigator>
    ) : (
      <Stack.Navigator>
        <Stack.Screen name="Login" options={{ headerShown: false }}>
          {(props) => (
            <Login
              {...props}
              onLoginSuccess={handleLoginSuccess}
              onAdminLogin={handleAdminLogin}
            />
          )}
        </Stack.Screen>
        <Stack.Screen name="SignUp" component={SignUp} />
      </Stack.Navigator>
```

```
      )}
      <StatusBar style="auto" />
    </NavigationContainer>
  );
}
```

Login.js

```
import React, { useState, useRef, useEffect } from 'react';
import { View, TextInput, Text, StyleSheet, SafeAreaView, KeyboardAvoidingView,
TouchableOpacity, Image, Platform, ScrollView, Animated } from 'react-native';
import axios from 'axios';
import Icon from 'react-native-vector-icons/Ionicons';
import AsyncStorage from '@react-native-async-storage/async-storage';
// Import Ionicons for eye icon

const Login = ({ navigation, onLoginSuccess, onAdminLogin }) => {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');
  const [isUsernameFocused, setIsUsernameFocused] = useState(false);
  const [isPasswordFocused, setIsPasswordFocused] = useState(false);
  const [isPasswordVisible, setIsPasswordVisible] = useState(false); // State for
password visibility

  const usernameLabelAnim = useRef(new Animated.Value(0)).current;
  const passwordLabelAnim = useRef(new Animated.Value(0)).current;

  useEffect(() => {
    // Username label animation
    Animated.timing(usernameLabelAnim, {
      toValue: isUsernameFocused || username !== '' ? 1 : 0,
      duration: 200,
      useNativeDriver: false,
    }).start();
  }, [isUsernameFocused, username]);

  useEffect(() => {
    // Password label animation
    Animated.timing(passwordLabelAnim, {
      toValue: isPasswordFocused || password !== '' ? 1 : 0,
      duration: 200,
      useNativeDriver: false,
    }).start();
  }, [isPasswordFocused, password]);

  const handleLogin = async () => {
    if (!username || !password) {
      setError('Username and Password are required.');
```

```
      return;
    }

    try {
      const response = await axios.post('https://boss-turkey-happily.ngrok-
free.app/login', { username, password });
      const user = response.data;

      // Store the username in AsyncStorage
      if (user.username) {
        await AsyncStorage.setItem('username', user.username);
      } else {
        console.error('Username not found in response');
      }

      if (user.isAdmin) {
        onAdminLogin();
      } else {
        onLoginSuccess();
      }
    } catch (err) {
      console.error('Login error:', err);
      setError('Login failed. Please check your credentials.');
    }
  };

  const gotoRegister = () => {
    navigation.navigate('SignUp');
  };

  const labelStyle = (labelAnim) => ({
    position: 'absolute',
    left: 15,
    top: labelAnim.interpolate({
      inputRange: [0, 1],
      outputRange: [12, -25],
    }),
    fontSize: labelAnim.interpolate({
      inputRange: [0, 1],
      outputRange: [15, 12],
    }),
    color: '#075eec',
  });

  return (
    <SafeAreaView style={styles.safeArea}>
      <KeyboardAvoidingView
        behavior={Platform.OS === 'ios' ? 'padding' : 'height'}
        style={styles.container}
```

```
        keyboardVerticalOffset={Platform.OS === 'ios' ? 60 : 0}
      >
        <ScrollView contentContainerStyle={styles.scrollContainer}>
          <View style={styles.innerContainer}>
            <View style={styles.header}>
              <Image source={require('../../assets/main_logo.png')}
style={styles.headerImg} />
              <Text style={styles.title}>Login</Text>
              <Text style={styles.subtitle}>Welcome To We-Together</Text>
            </View>
            <View style={styles.formContainer}>
              <View style={styles.form}>
                <View style={styles.input}>
                  {/* Username Input */}
                  <Animated.Text style={labelStyle(usernameLabelAnim)}>
                    Username:
                  </Animated.Text>
                  <TextInput
                    autoCapitalize='none'
                    autoCorrect={false}
                    keyboardType='email-address'
                    style={styles.inputText}
                    placeholder={isUsernameFocused || username !== '' ? '' :
'Username'}
                    placeholderTextColor="#003f5c"
                    onFocus={() => setIsUsernameFocused(true)}
                    onBlur={() => setIsUsernameFocused(false)}
                    onChangeText={(text) => setUsername(text)}
                  />
                </View>
                <View style={styles.input}>
                  {/* Password Input */}
                  <Animated.Text style={labelStyle(passwordLabelAnim)}>
                    Password:
                  </Animated.Text>
                  <View style={styles.passwordContainer}>
                    <TextInput
                      secureTextEntry={!isPasswordVisible} // Change visibility based
on state
                      style={styles.inputText}
                      placeholder={isPasswordFocused || password !== '' ? '' :
'Password'}
                      placeholderTextColor="#003f5c"
                      onFocus={() => setIsPasswordFocused(true)}
                      onBlur={() => setIsPasswordFocused(false)}
                      onChangeText={(text) => setPassword(text)}
                    />
                    <TouchableOpacity onPress={() =>
setIsPasswordVisible(!isPasswordVisible)} style={styles.eyeIconContainer}>
```

```
                    <Icon name={isPasswordVisible ? 'eye-off' : 'eye'} size={24}
color="#075eec" />
                  </TouchableOpacity>
                </View>
              </View>
              {error ? <Text style={styles.errorText}>{error}</Text> : null}
              <View style={styles.formAction}>
                <TouchableOpacity onPress={handleLogin}>
                  <View style={styles.btn}>
                    <Text style={styles.btnText}>Login</Text>
                  </View>
                </TouchableOpacity>
              </View>
            </View>
          </View>
          <View style={styles.regBtn}>
            <Text style={styles.hehe}>Don't have an account?</Text>
            <Text style={styles.regText} onPress={gotoRegister}>Sign Up</Text>
          </View>
        </View>
      </ScrollView>
    </KeyboardAvoidingView>
  </SafeAreaView>
 );
};

const styles = StyleSheet.create({
 safeArea: {
  flex: 1,
  backgroundColor: '#fff',
 },
 scrollContainer: {
  flexGrow: 1,
  position:'static',
 },
 innerContainer: {
  flex: 1,
  padding: 24,
 },
 header: {
  alignItems: 'center',
  marginTop: 90,
 },
 headerImg: {
  width: 300,
  height: 80,
  marginBottom: 20,
 },
 title: {
```

```
    fontSize: 27,
    color: 'black',
    marginBottom: 6,
    textAlign: 'center',
    fontFamily: 'Poppins-Bold',
  },
  subtitle: {
    textAlign: 'center',
    color: 'gray',
    fontFamily: 'Poppins-SemiBold',
  },
  formContainer: {
    flex: 1,
    justifyContent: 'center',
    top: 80,
  },
  form: {
    marginBottom: 150,
  },
  input: {
    marginBottom: 28,
    position: 'relative',
    paddingTop: 0,
    paddingBottom: 5,
  },
  passwordContainer: {
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'space-between',
    position: 'relative', // Set position to relative for absolute positioning of icon
  },
  inputText: {
    backgroundColor: '#ebecf4',
    height: 45,
    paddingHorizontal: 15,
    borderRadius: 12,
    fontSize: 15,
    fontFamily: 'Poppins-Normal',
    color: '#222',
    flex: 1,
  },
  eyeIconContainer: {
    position: 'absolute',
    right: 15, // Adjust position as needed
    top: 10, // Center the icon vertically
  },
  btn: {
    backgroundColor: '#075eec',
    borderRadius: 50,
```

```
    alignSelf: 'center',
    paddingVertical: 10,
    width: 150,
  },
  formAction: {
    marginVertical: 24,
  },
  btnText: {
    fontSize: 16,
    color: '#fff',
    textAlign: 'center',
    fontFamily: 'Poppins-Bold',
    top: 2,
  },
  regBtn: {
    alignItems: 'center',
    marginBottom: 24,
  },
  regText: {
    fontSize: 15,
    textDecorationLine: 'underline',
    fontFamily: 'Poppins-Bold',
  },
  hehe: {
    fontFamily: 'Poppins-Normal',
  },
  errorText: {
    color: 'red',
    marginVertical: 10,
    textAlign: 'center',
  },
});

export default Login;
```

4.2 Screenshots

Sign Up and Login Page:

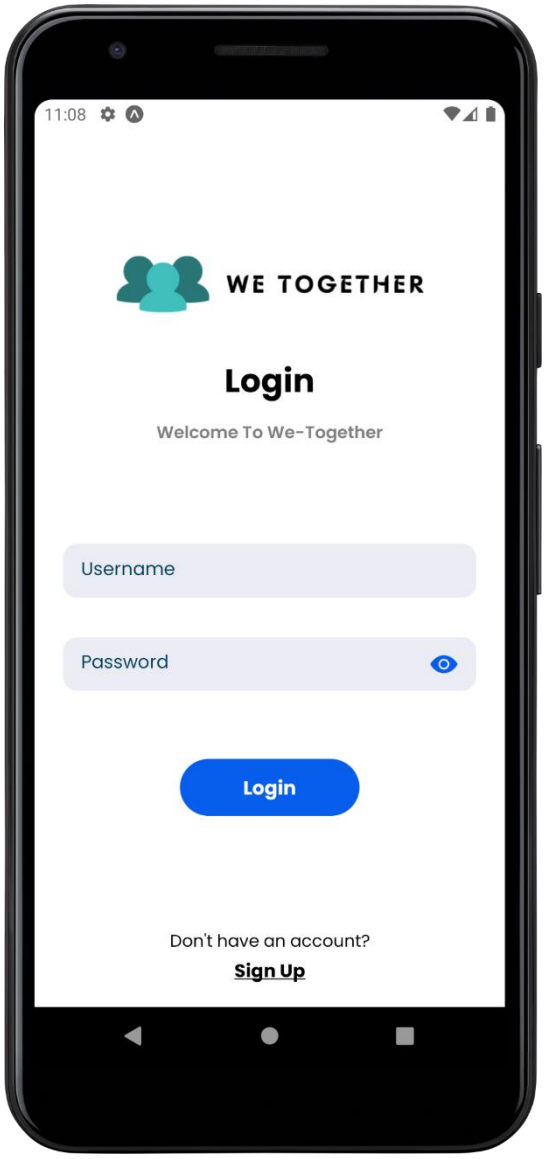

Fig 4.1 Login Page

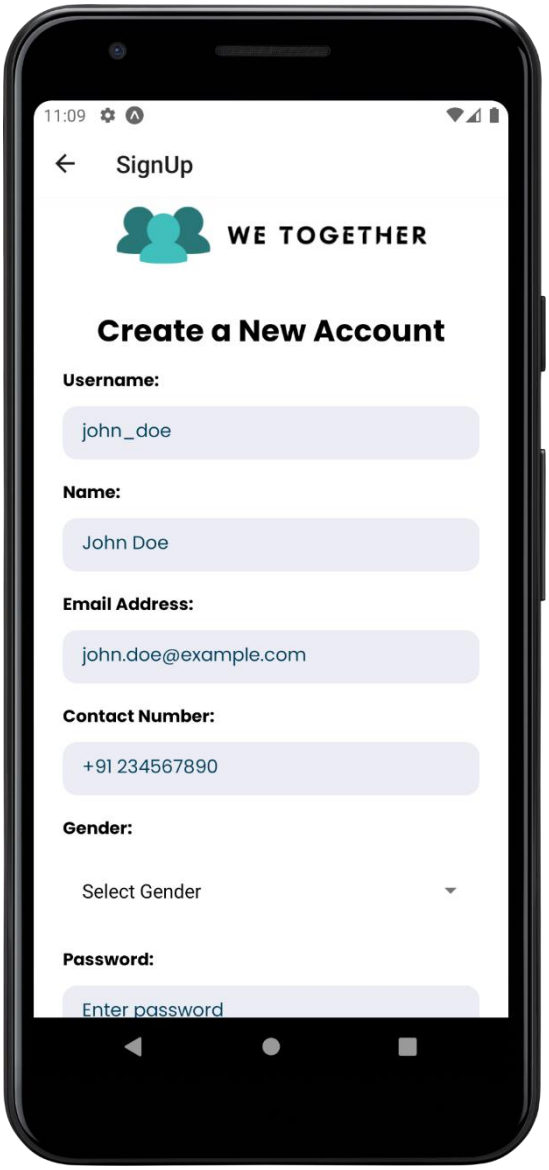

Fig 4.2 Sign Up Page

Home Page and Navigation Page:

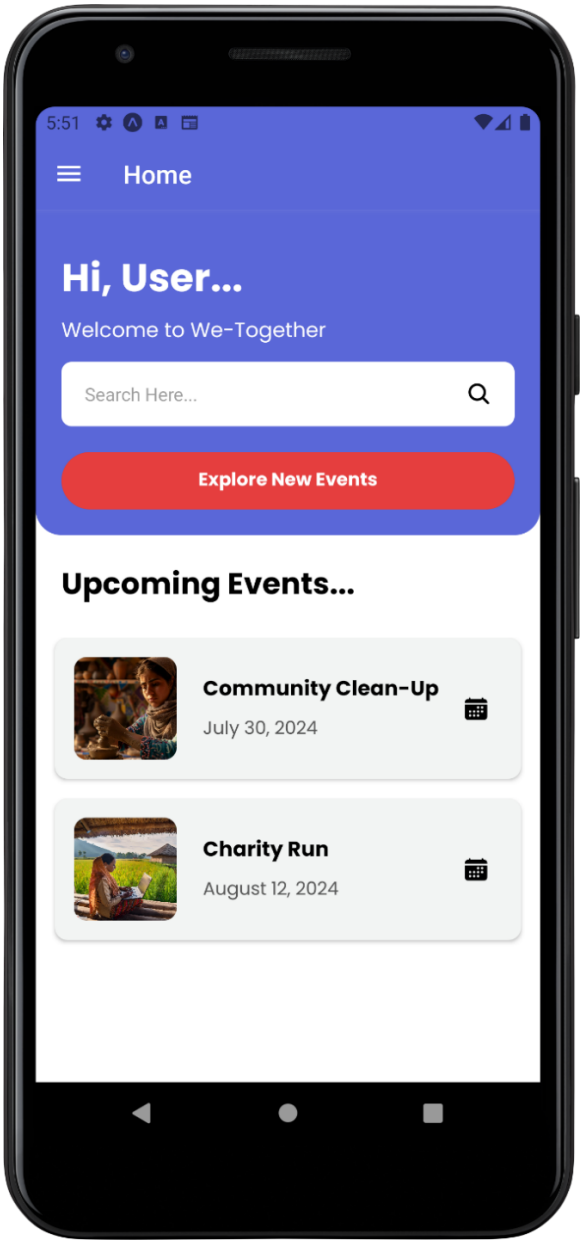

Fig 4.3 Home Screen page

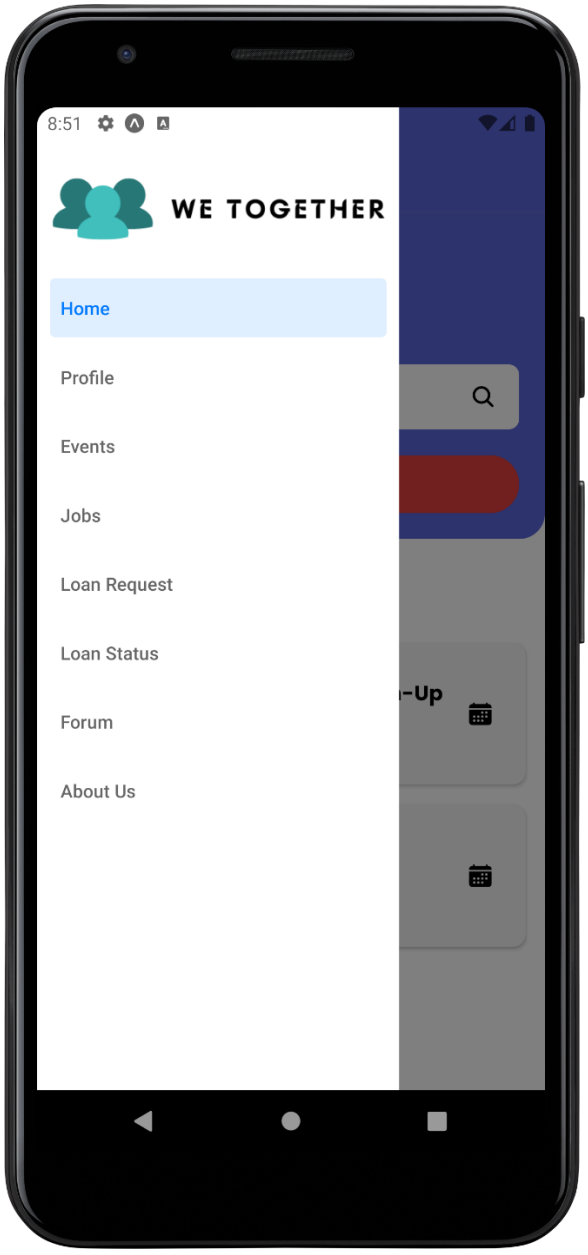

Fig 4.4 Navigation Page

Event Page and Event Registration Page:

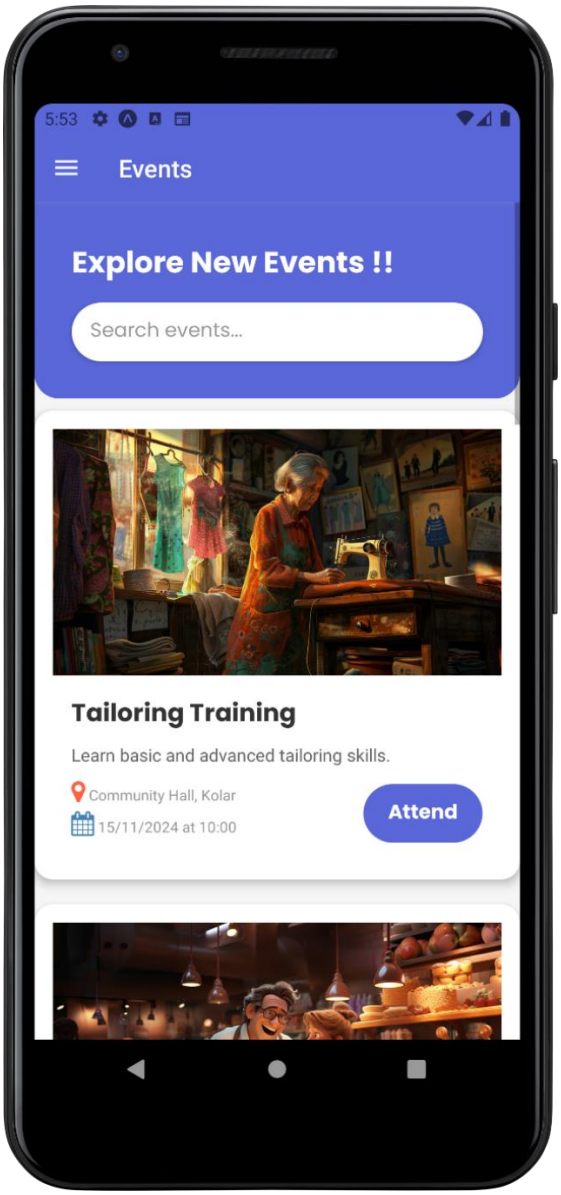

Fig 4.5 Events page

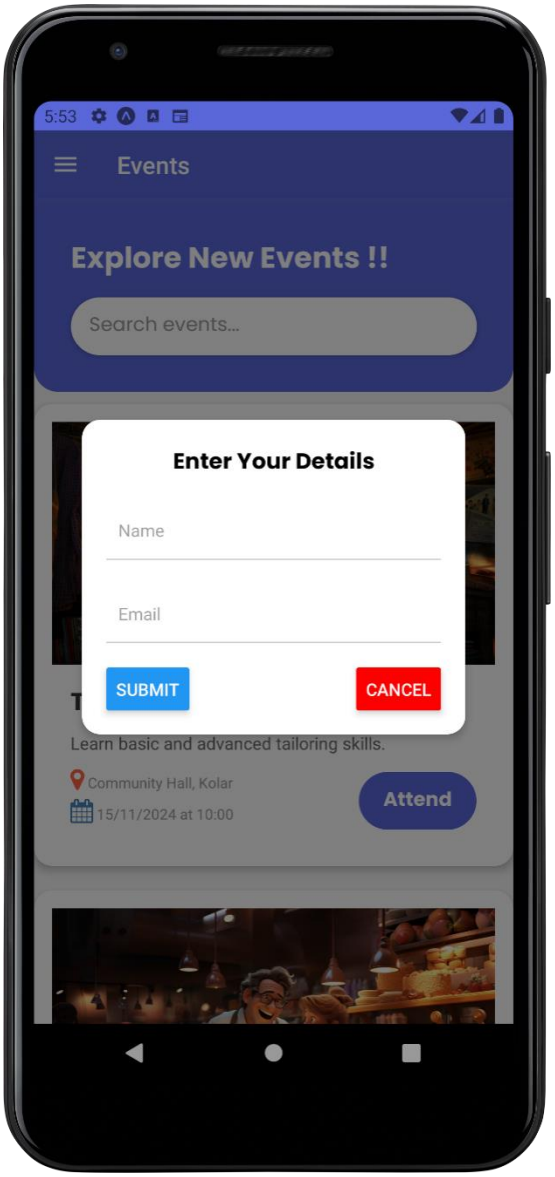

Fig 4.6 Event Registration Page

Job Page and Job Application Page:

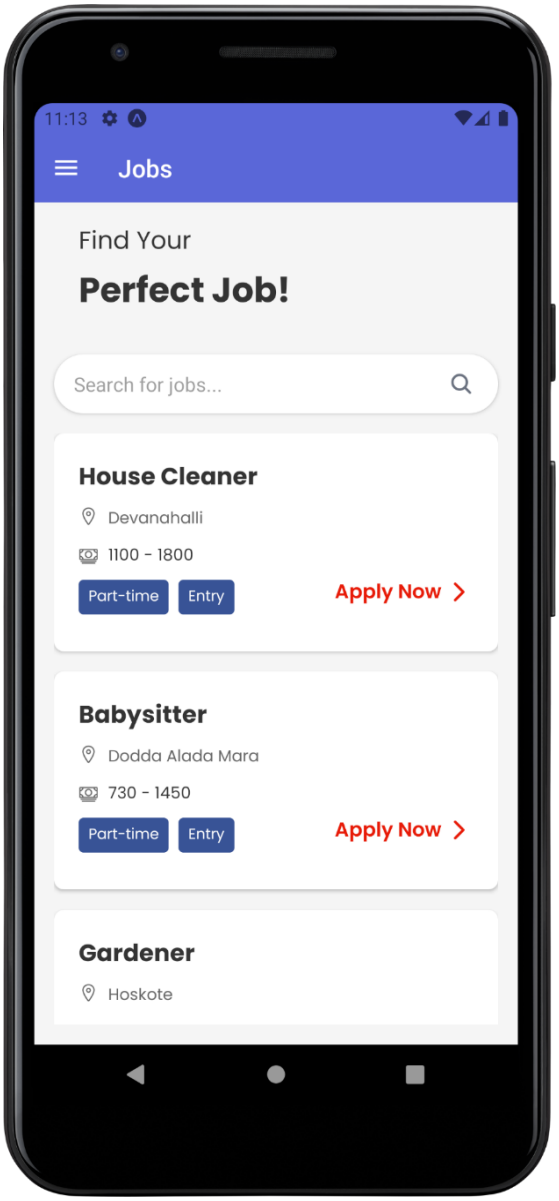

Fig 4.7 Jobs page

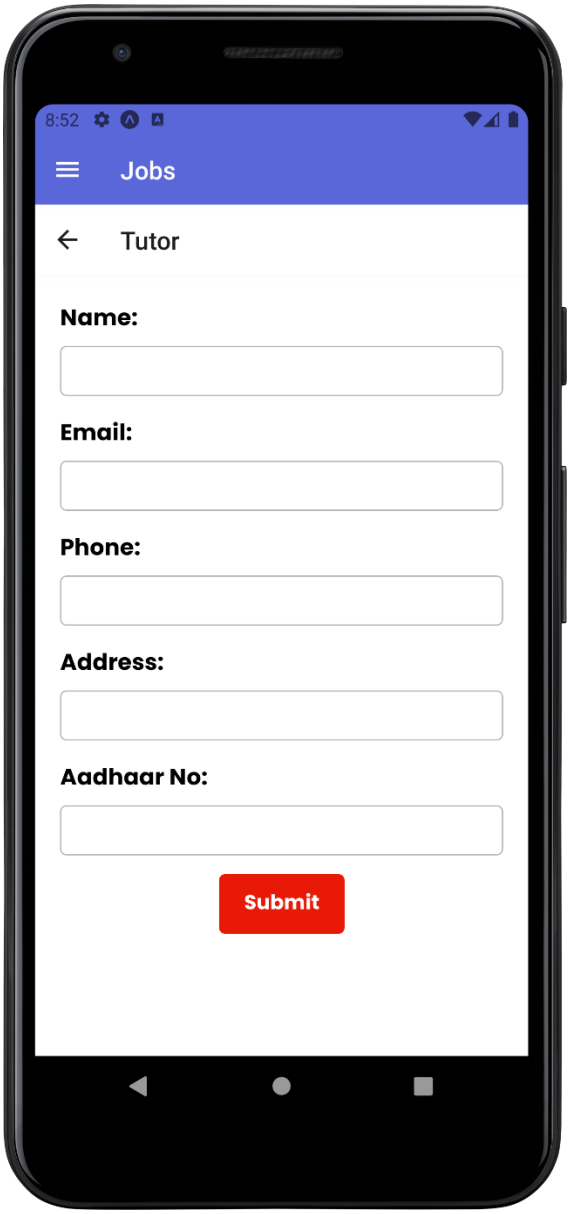

Fig 4.8 Job Application Page

Loan Page and Loan Status Page:

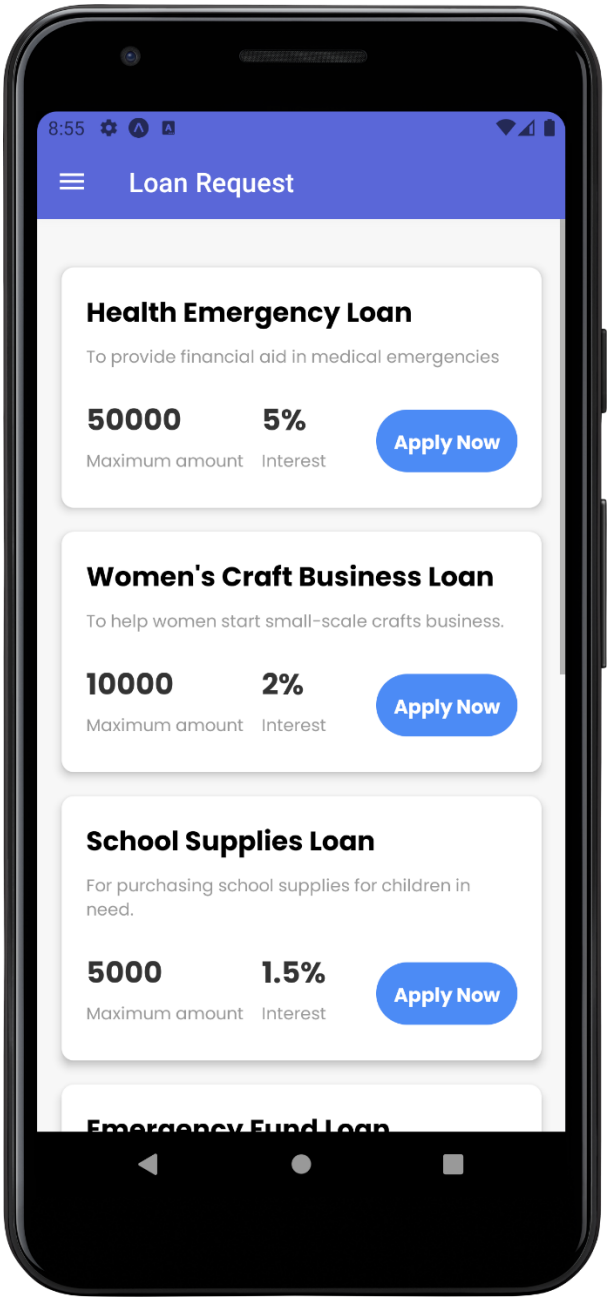

Fig 4.9 Loan page

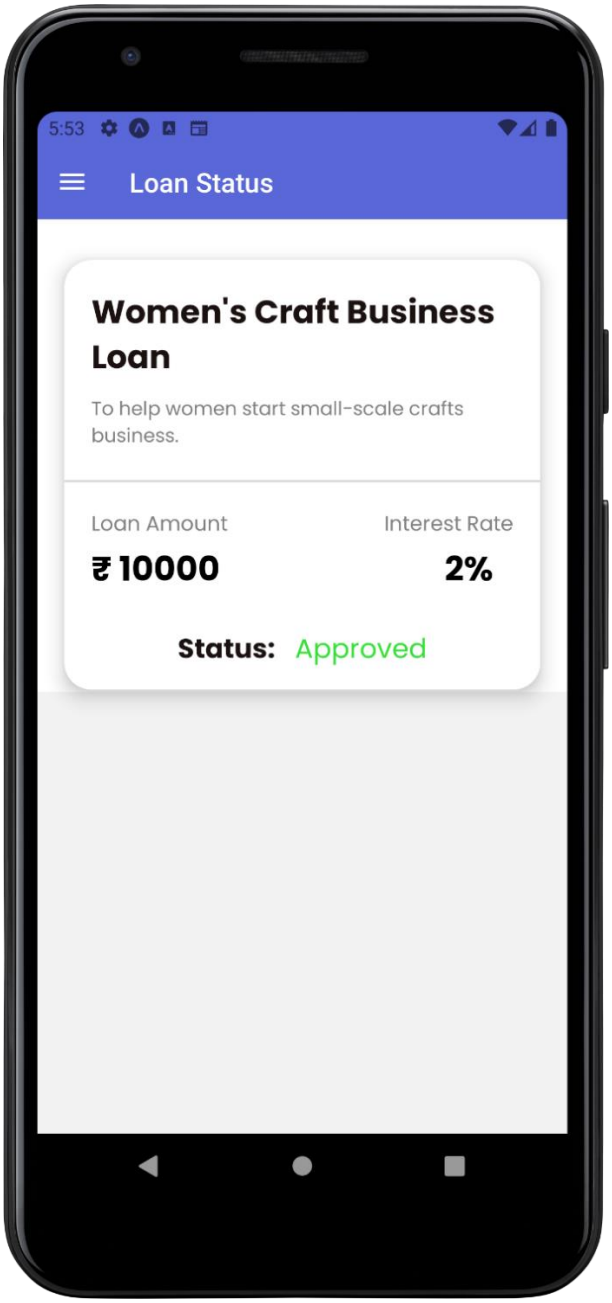

Fig 4.10 Loan Status Page

Forum Page and Profile Page:

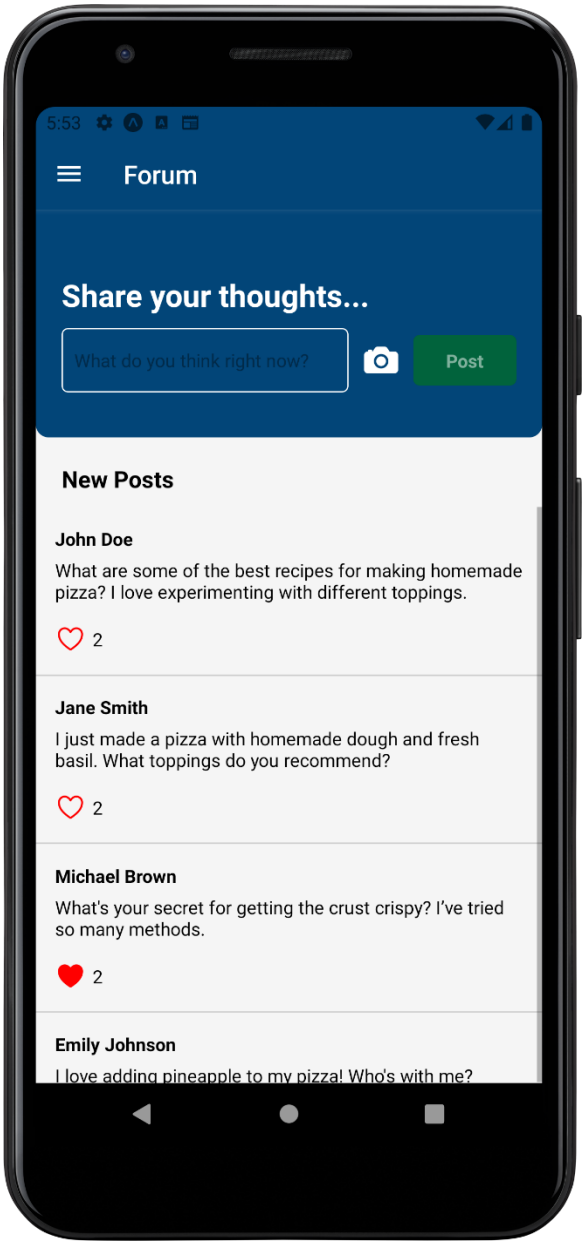

Fig 4.11 Forum page

Fig 4.12 Profile Page

Admin Home Page and Create Admin Page:

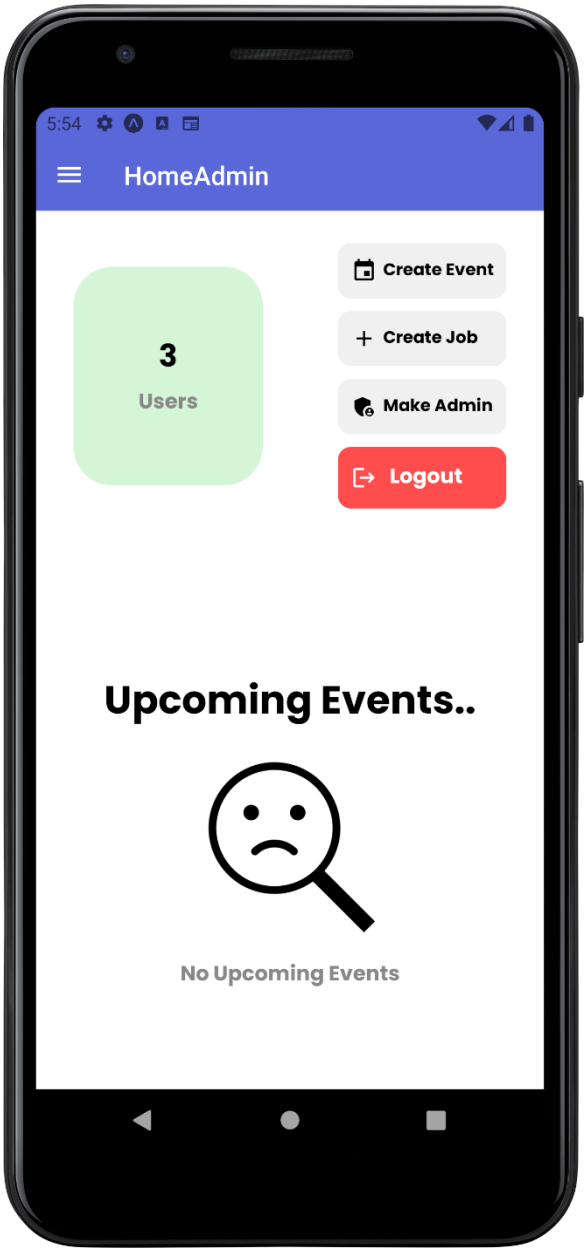

Fig 4.13 Admin home Page

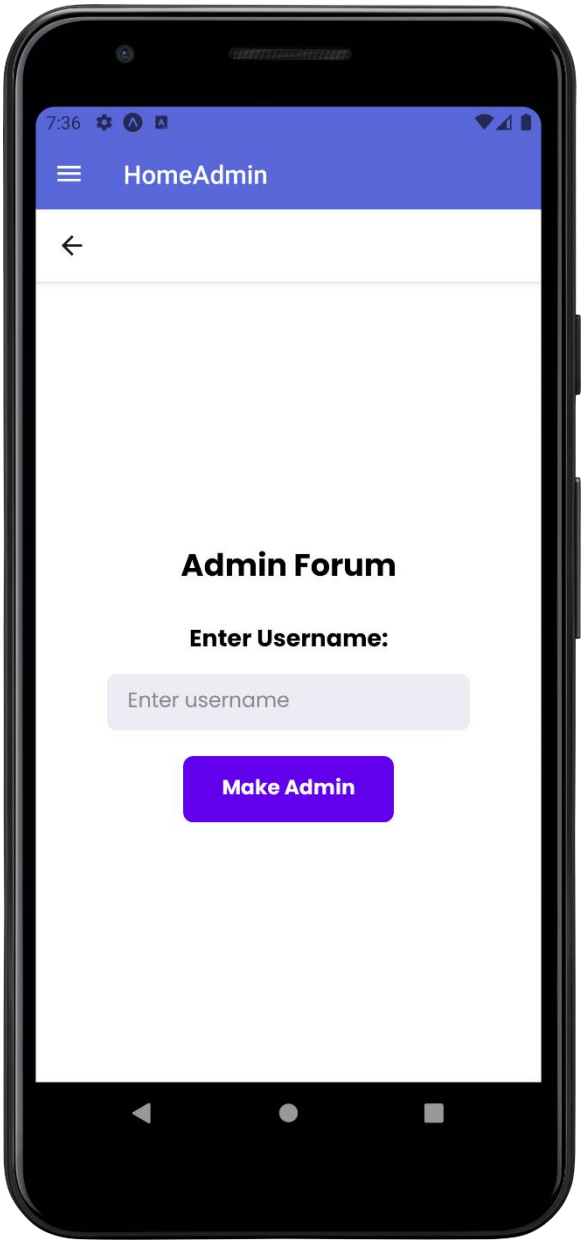

Fig 4.14 Admin home Page

Admin Event Page and Event Attendance Page:

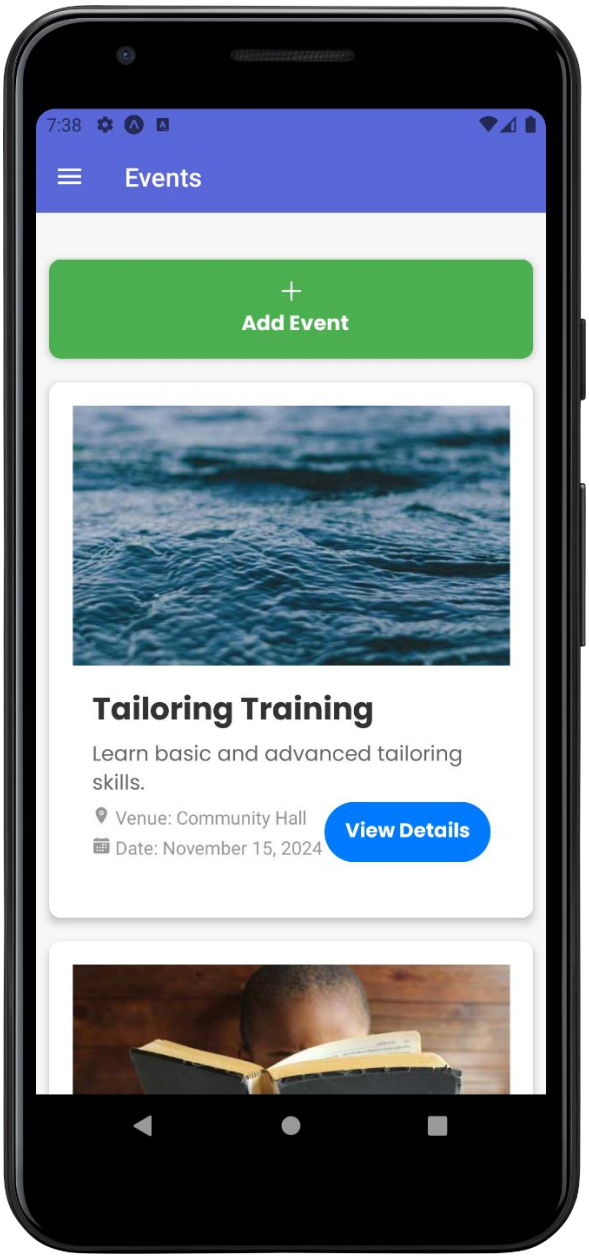

Fig 4.15 Admin event Page

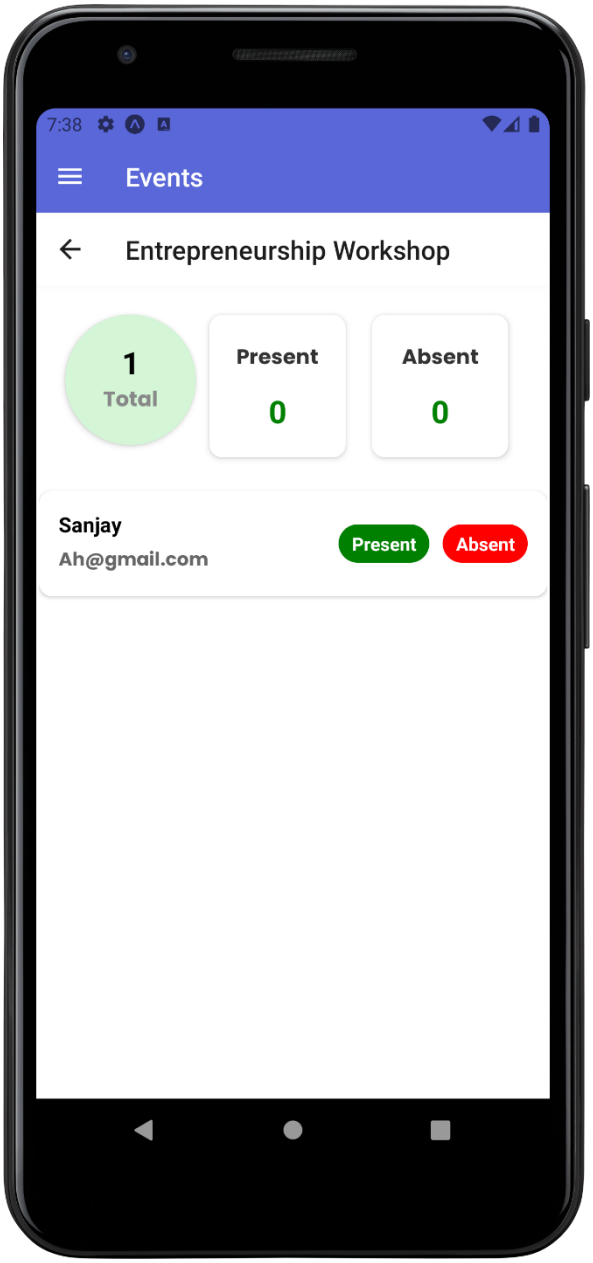

Fig 4.16 Admin attendance Page

Admin Job Page and Applicants Page:

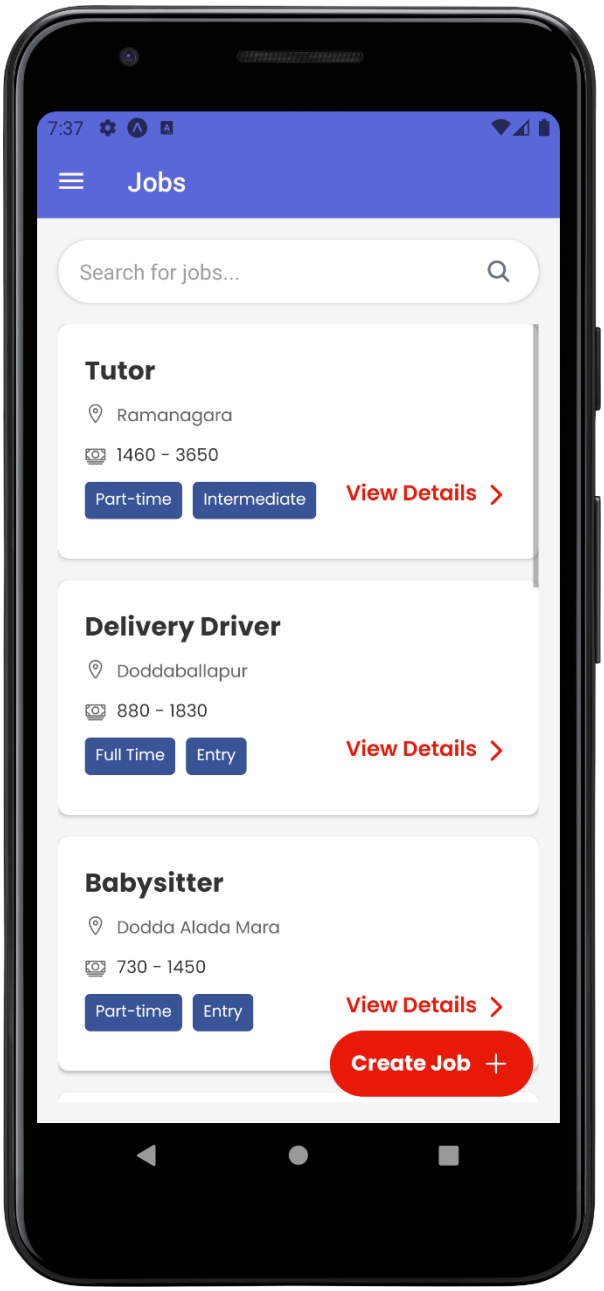

Fig 4.17 Admin event Page

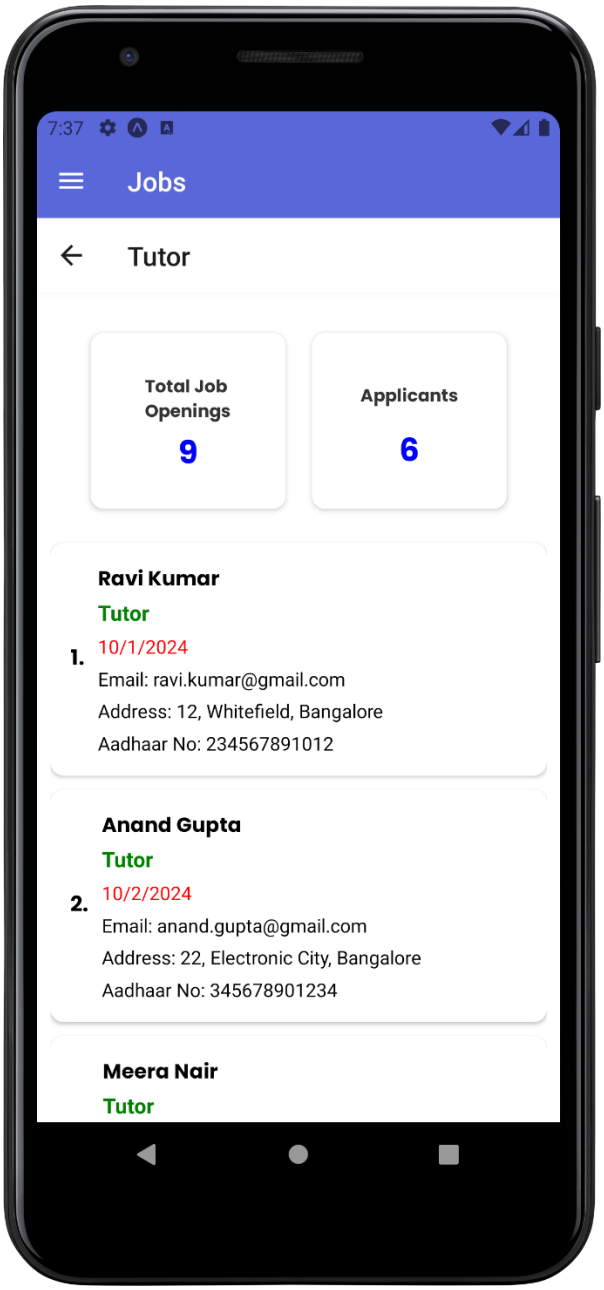

Fig 4.18 Admin attendance Page

Admin Loan Page and Loan Approval Page:

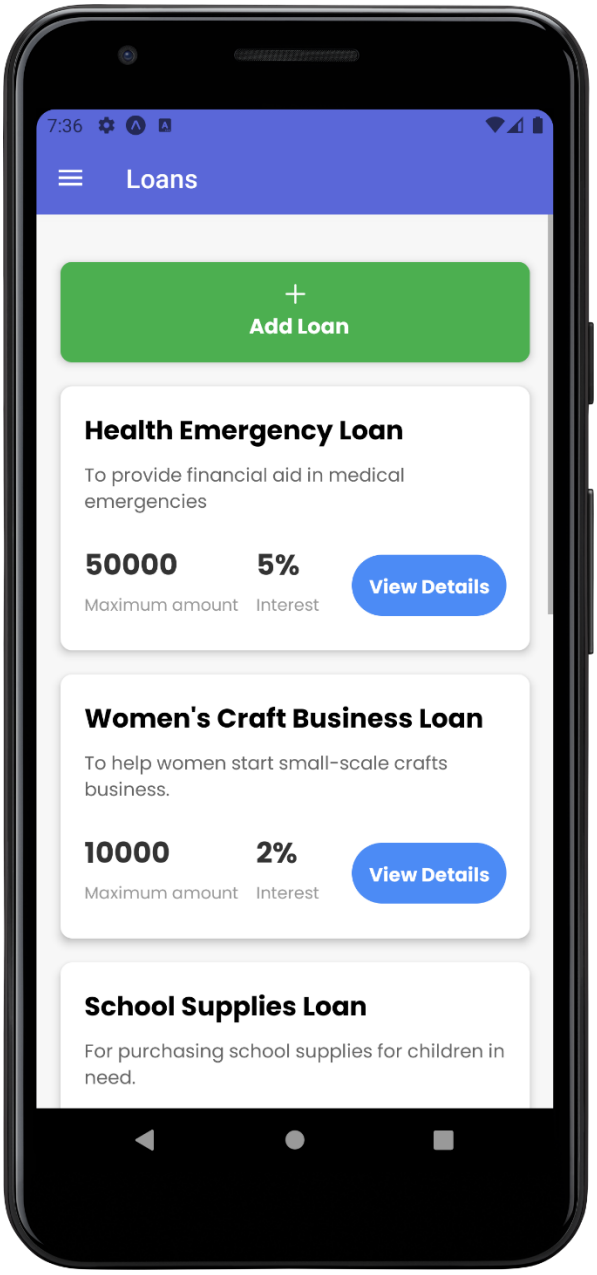

Fig 4.19 Loan Page

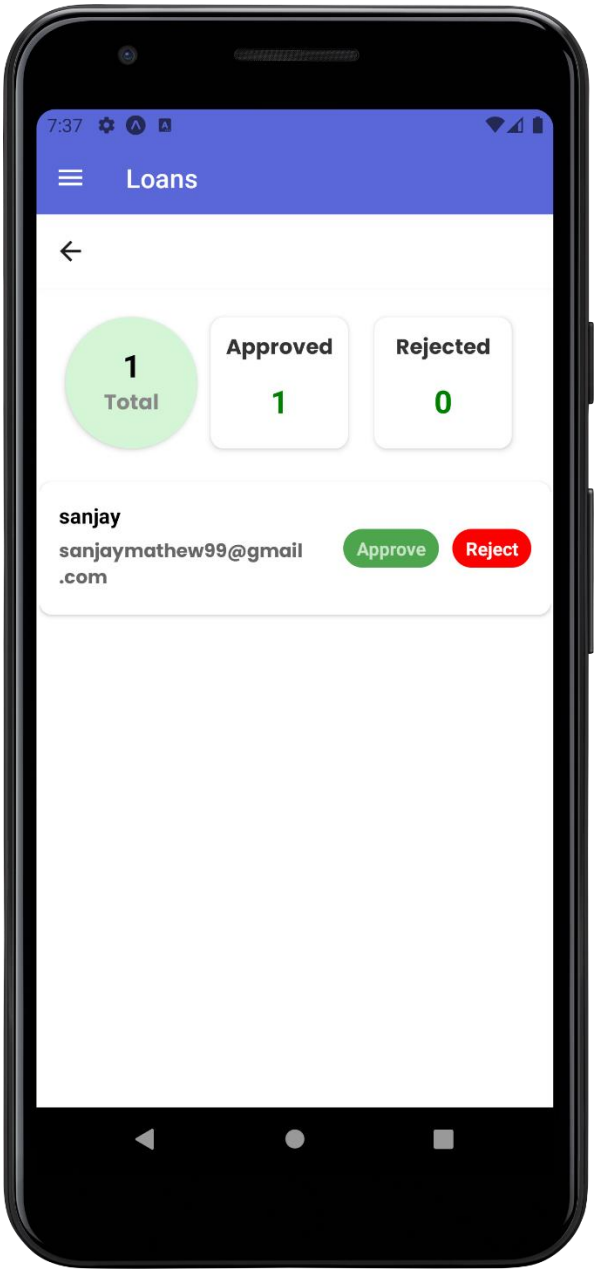

Fig 4.20 Loan Approval Page

Admin Job Creation and Loan Creation Page:

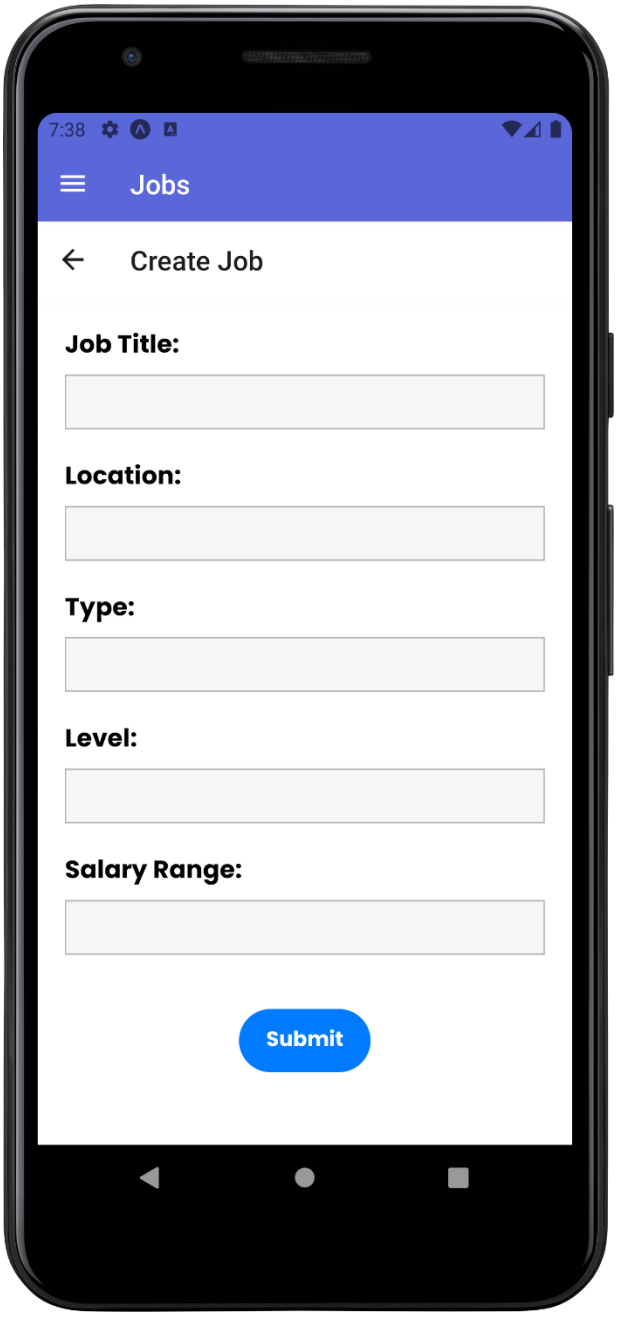

Fig 4.21 Job Creation Page

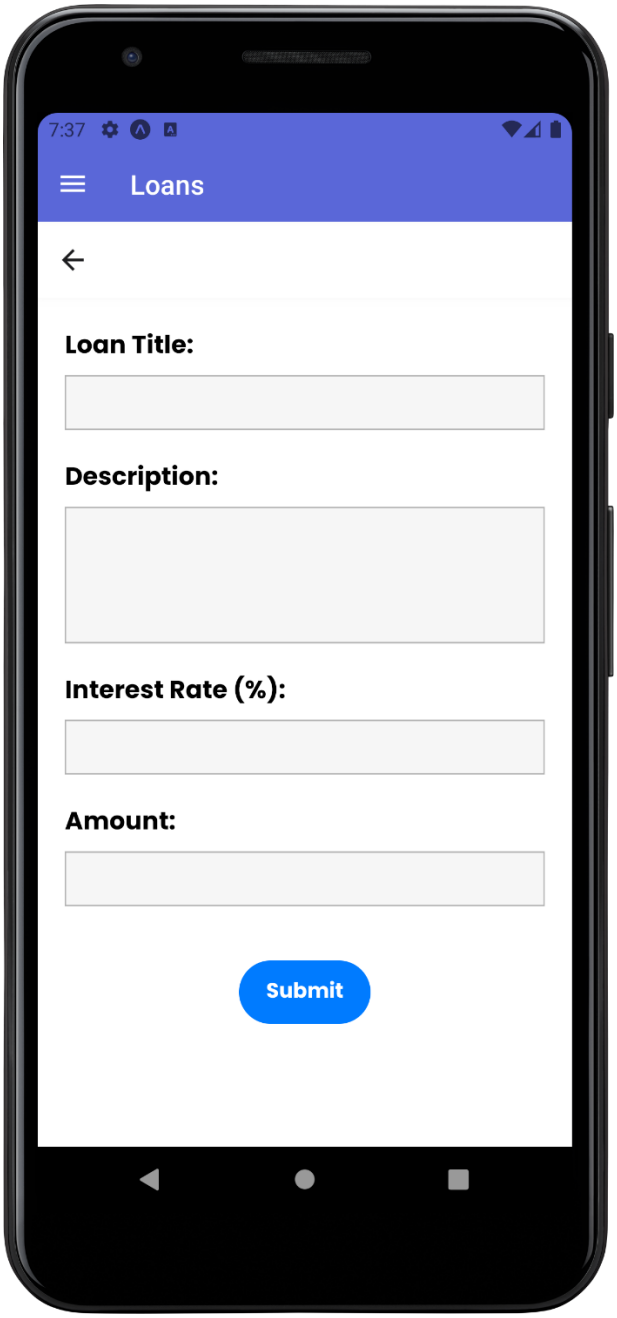

Fig 4.22 Loan Forum Page

Database Screenshots

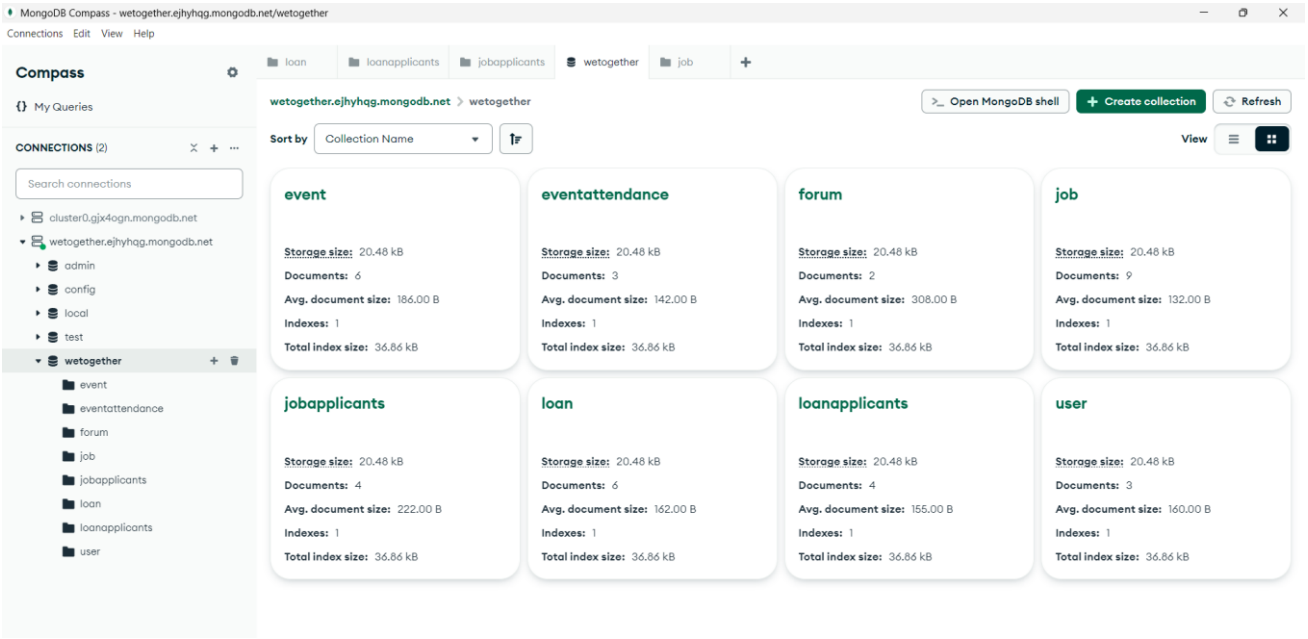

Fig 4.23 MongoDB Database We Together

User Structure:

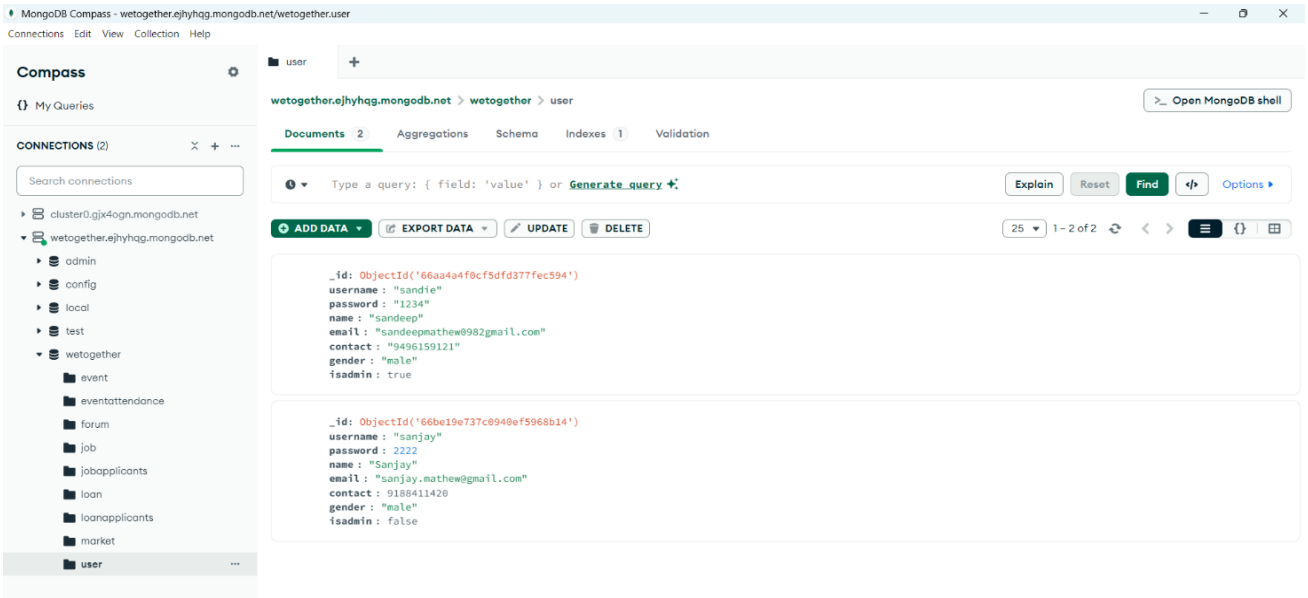

Fig 4.24 User Structure

Events Structure:

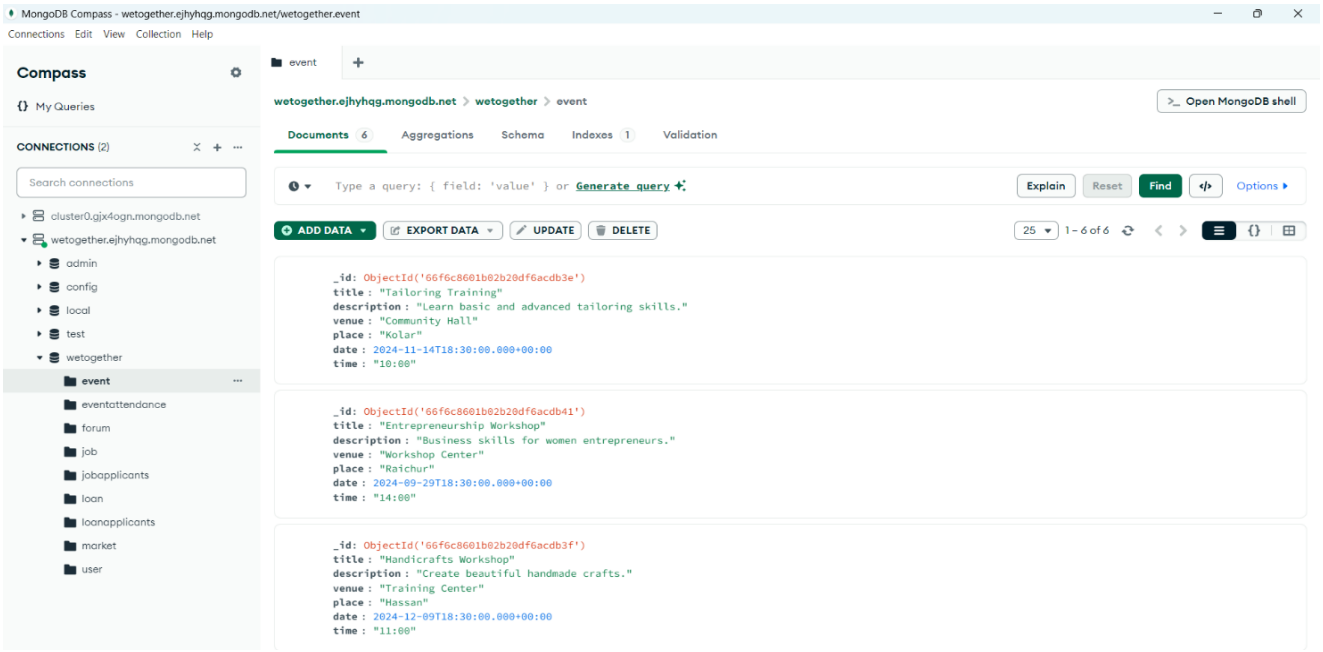

Fig 4.25 Events Structure

Event Attendance Structure:

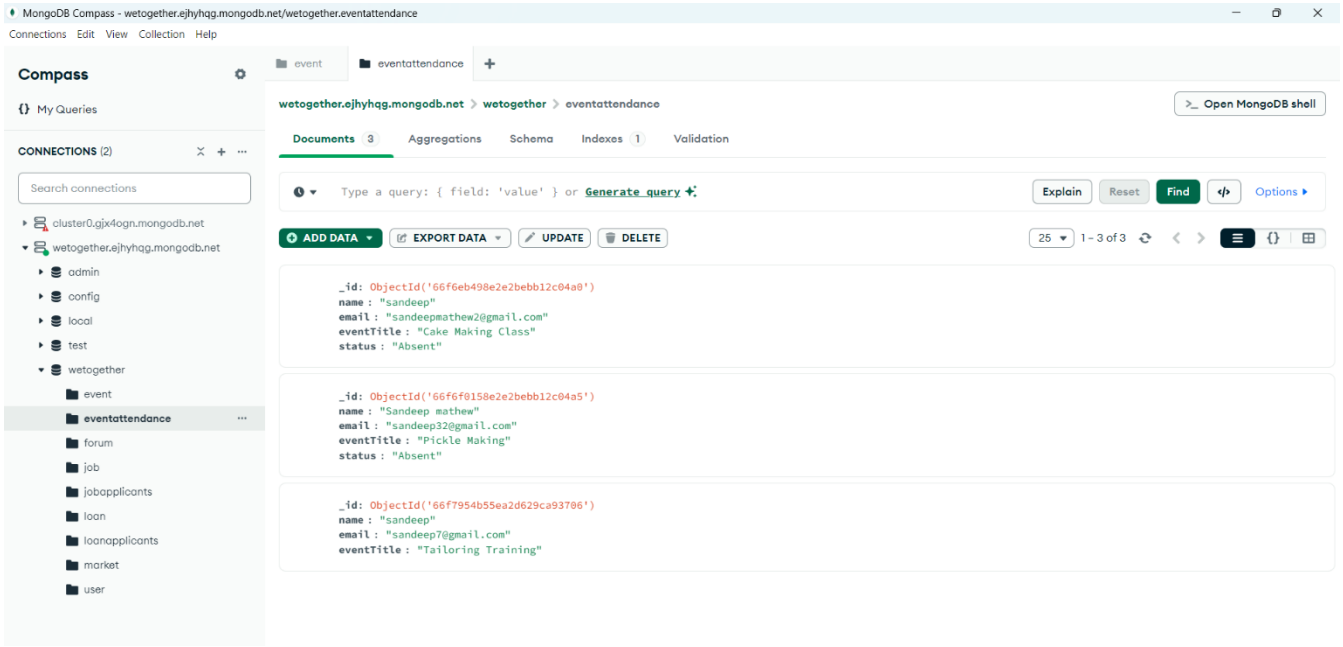

Fig 4.26 Event Attendance Structure

Jobs Structure:

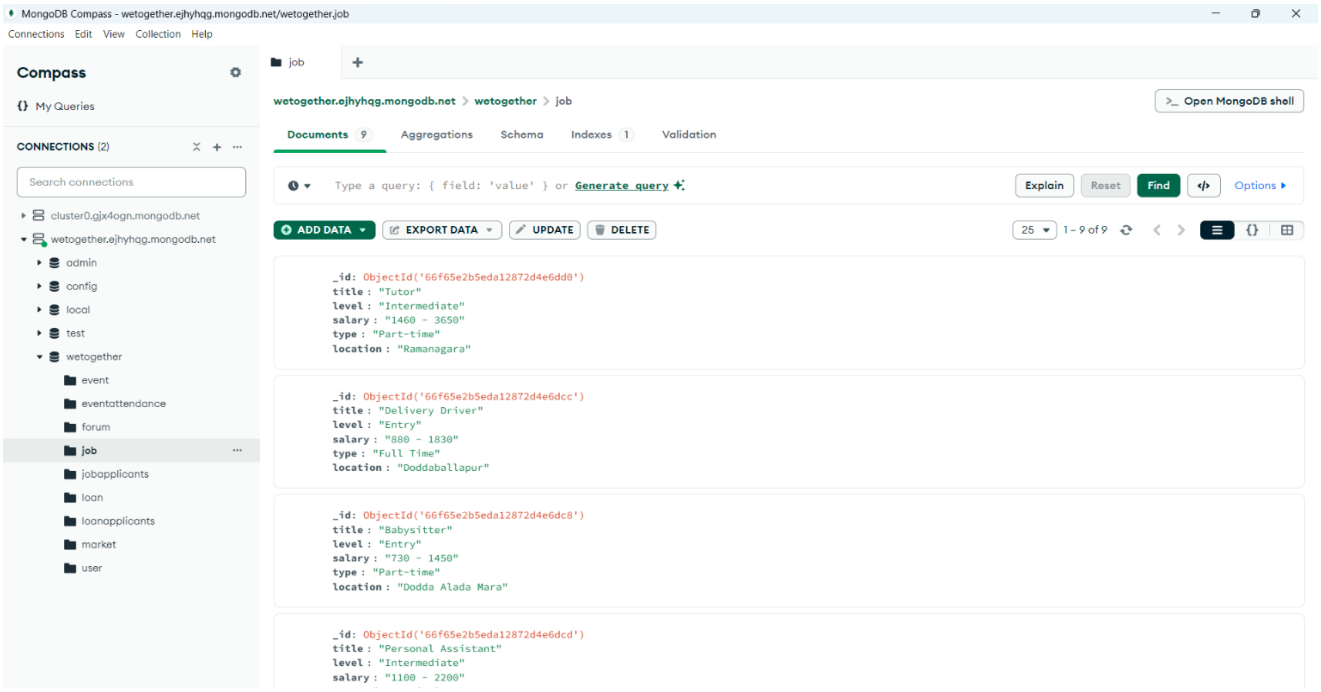

Fig 4.27 Jobs Structure

Job Applicants Structure:

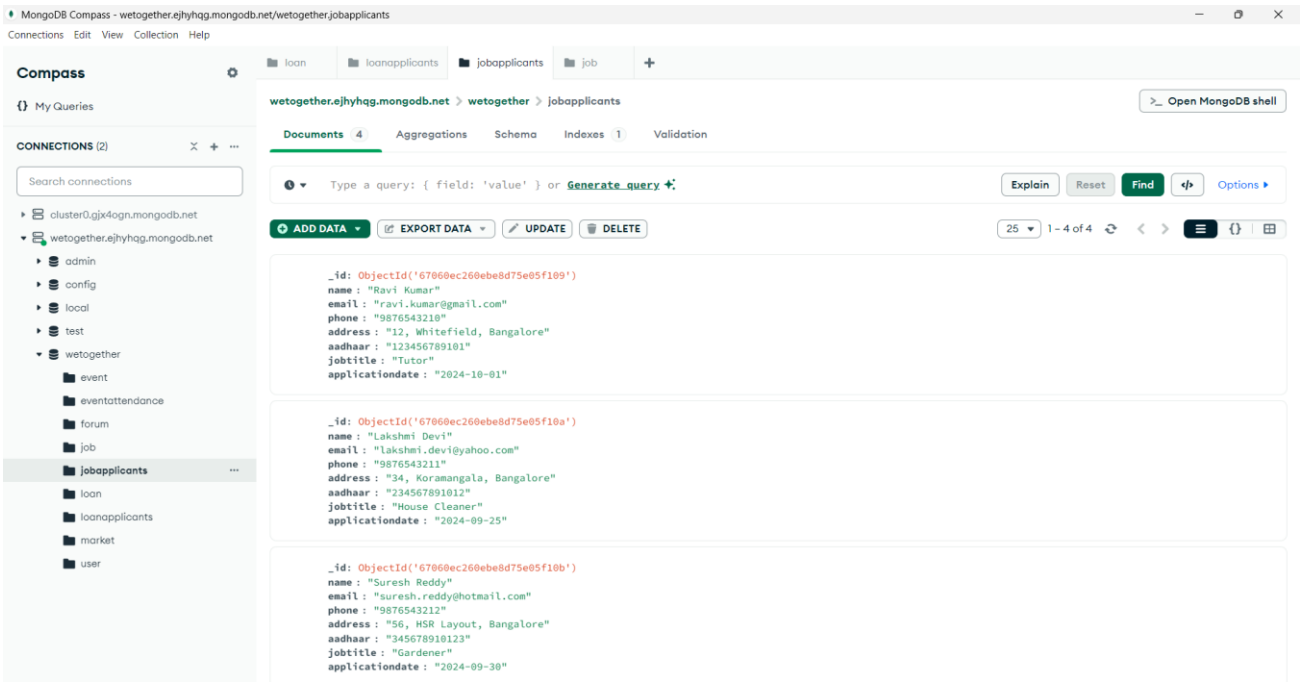

Fig 4.28 Job Applicants Structure

Loan Structure:

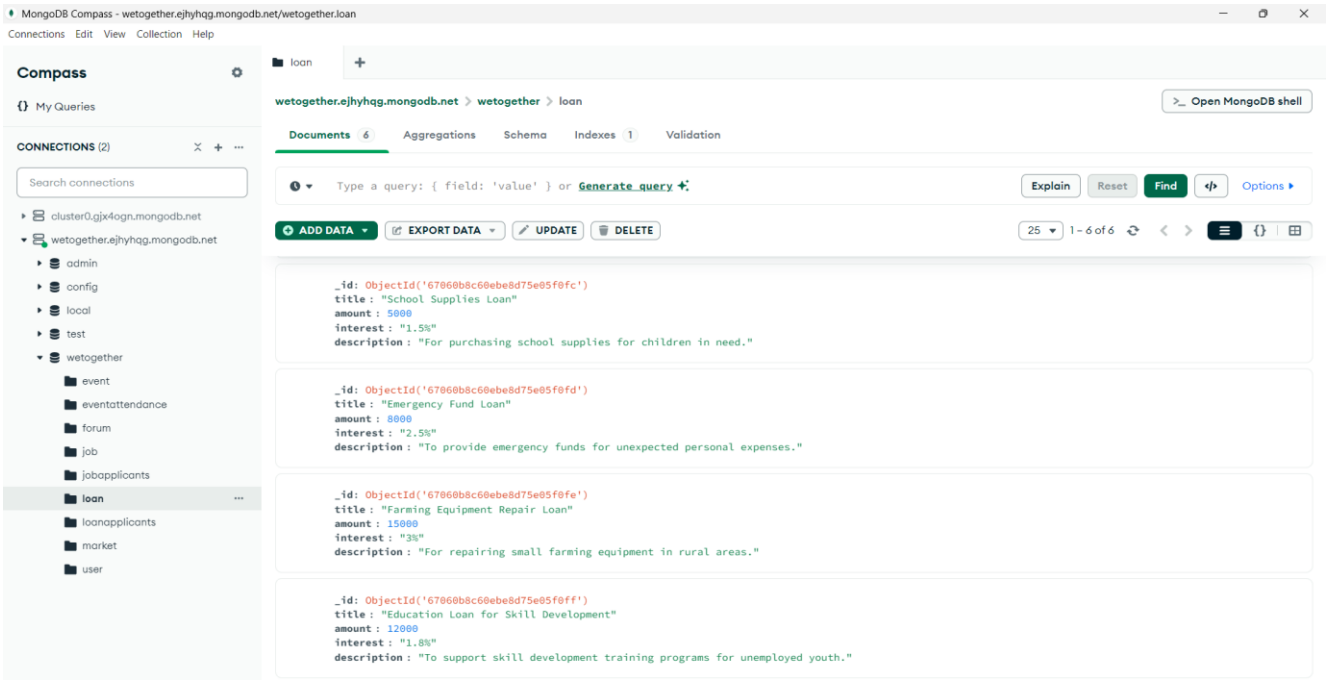

Fig 4.29 Loan Structure

Loan Applicants Structure:

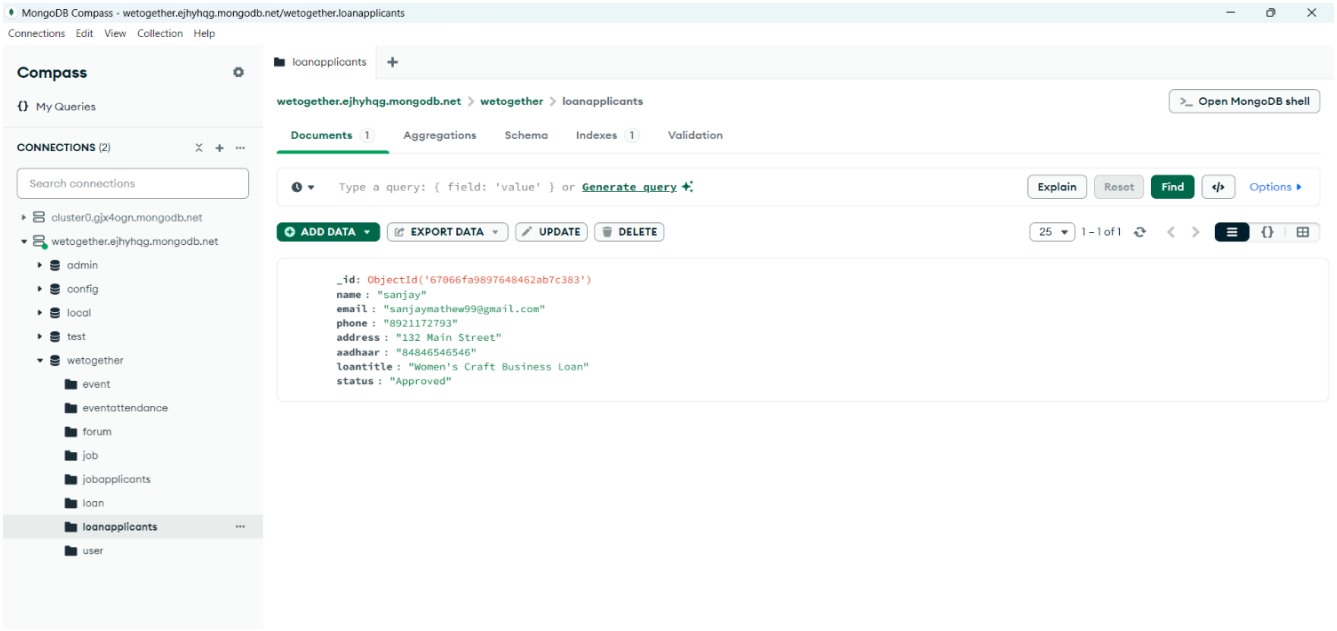

Fig 4.30 Loan Applicants Structure

Forum Structure:

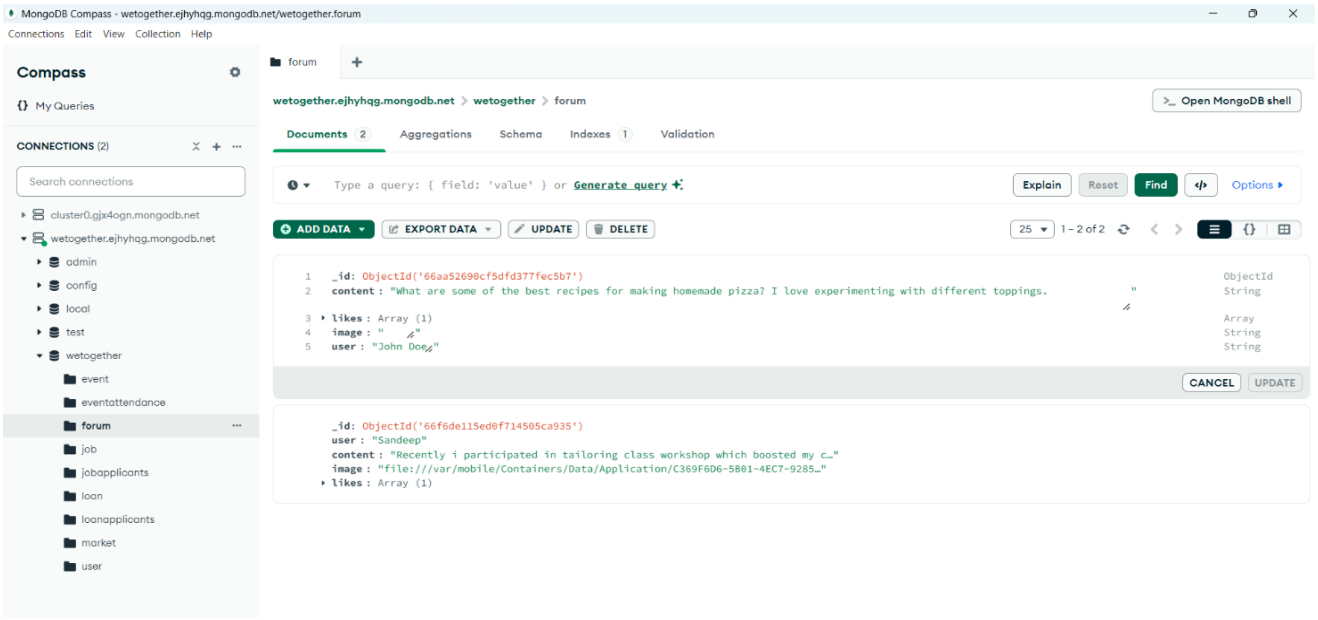

Fig 4.31 Forum Structure

5.TESTING

5.1 METHODS OF TESTING

A software testing strategy incorporates software test cases into a sequence of meticulously organized stages that culminate in the effective development of software.

Verification and validation are more broadly defined as software testing. The set of procedures known as verification makes sure that the developed software can be linked back to the needs of theclient.

The steps involved in testing are-

5.1.1 Unit Testing: Testing each design unit independently is known as unit testing. In this project, we independently tested each design unit to ensure that there no mistakes. For this testing, every design is executed separately. If an error arises after each page is executed, the rectification method is completed immediately.

5.1.2 Integration testing: In the project, combined many units of modules to form a sub- system. These sub-systems are then tested. This is done to see whether the modules can be integrated properly. Based on integration testing some necessary changes were made to the design.

5.1.3 System testing: System testing is done to ensure the entire software performs its function as intended. In our project all the tested subsystems were integrated and tested for all the possible ranges of coupling variables, based on the testing errors were rectified for a pleasant working experience.

5.1.4 Acceptance testing: The goal of acceptance testing is to see if the software meets all the requirements as needed. The testing was performed by data of all the users of the system. It was found that the software meets all the requirements of the host, tenant, administrator, and service providers as needed.

5.2 Test Cases and Reports

Login Module

5.2.1.1	**Input**: Valid Credentials with User Type	**Output**: Login Successful	User logged into account and redirected to home page
5.2.1.2	**Input**: Valid Credentials with blank fields	**Output**: Please Fill all the details	User is shown an error message that credentials are empty
5.2.1.3	**Input**: Invalid Credentials	**Output**: Invalid Email or Password	User is shown an error message that credentials are wrong

Registration Module

5.2.2.1	**Input**: Valid Credentials with User Type	**Output**: Registration Successful	User registered and redirected to login page
5.2.2.2	**Input**: Valid **Credentials** with blank fields	**Output**: Please Fill all the details	User is shown an error message that credentials are empty
5.2.2.3	**Input**: Invalid Credentials	**Output**: Invalid Data Entered	User is shown an error message that credentials are wrong

Job Management Module

5.2.3.1	**Input**: Admin Creates Job (Name, Description, Time and Date, Location)	**Output**: Job Created Successfully	Once created, a confirmation message is shown
3.2	**Input:** Admin Creates Job with Missing Data	**Output:** Please fill in all required fields	An error message prompts the admin to enter all necessary job details
3.3	**Input:** View Created Jobs	**Output:** List of Created Jobs	Admins can view all created jobs in a structured list format, with relevant details

Loan Management Module

4.1	**Input**: View Loan Details	**Output**: Loan Information Displayed	The page displays details of users who took loans, including the loan amount, due date, and total amount lent by app
4.2	**Input**: View Loan Details with No Loans Available	**Output**: "No Loan Records Found"	An error message is shown when there are no loans to display

Attendance Management Module

5.1	**Input**: Mark Attendance for Event/Meeting	**Output**: Attendance Marked Successfully	Admins can mark attendance for community members during events or meetings, with a confirmation message
5.2	**Input**: Update Attendance for Non-Existent Event	**Output**: "Event Not Found"	An error message is displayed if the admin attempts to update attendance for an event that doesn't exist

Forum Module

6.1	**Input**: Post Topic/Experience	**Output**: Post Created Successfully	A success message confirms the post creation
6.2	**Input**: Like Post after Login	**Output**: Post Liked	The icon is changed to liked post icon

6.CONCLUSION

In conclusion, the "We Together" project represents a significant advancement in the pursuit of gender equality and economic empowerment. By establishing a dynamic self-help group (SHG) platform that prioritizes women's empowerment while including men, the initiative fosters a collaborative environment where all participants can thrive. The comprehensive modules, ranging from job and loan management to community engagement and training, address the pressing challenges faced by women in accessing resources and opportunities.

With its user-friendly design and practical functionalities, "We Together" not only simplifies the process of skill development and resource allocation but also ensures that all community members, regardless of their technological proficiency, can participate fully. This project is not just about enhancing economic independence; it is about building a supportive ecosystem that empowers individuals to break down barriers and achieve their potential.

Ultimately, "We Together" stands as a transformative initiative that aims to uplift communities, foster social cohesion, and contribute meaningfully to the realization of Sustainable Development Goals, particularly in promoting gender equality and innovation.

REFERENCES

[1] Kudumbashree Website < https://kudumbashree.org/>

[2] Lokos Website < https://lokos.in/#/ >